POLITECNICO DI TORINO

Master's Degree in COMMUNICATIONS AND COMPUTER NETWORKS ENGINEERING

Master's Degree Thesis

Implementation of standard-compliant ETSI ITS-G5 Networking and Transport Layers on ns-3

Supervisors

Prof. Claudio Ettore CASETTI

Dott. Marco MALINVERNO

Dott. Francesco RAVIGLIONE

Candidate

Carlos Mateo RISMA CARLETTI

APRIL 2021

Summary

The future transportation systems are expected to be revolutionized by connected and fully automated vehicles, significantly improving road safety and traffic efficiency, together with the overall user experience. These vehicular applications come at a high deployment cost and with a complexity that makes it necessary to be extensively tested and analyzed with simulation tools.

This thesis puts its basis on an open-source vehicular network simulation framework built on the ns-3 simulator and making use of SUMO (Simulation of Urban MObility) to manage the mobility. This framework, called MS-VAN3T, enables the creation of vehicular scenarios and models all the aspects of the communication among the various entities in the created scenarios. One of the key features of this framework that sets it apart from others is that it integrates multiple communication stacks under a single open-source repository. In this framework the message exchange is based on standardized ITS (Intelligent Transport Systems) messages, defined by ETSI (European Telecommunications Standards Institute) for the Facilities layer, more specifically CAM (Cooperative Awareness Message) and DENM (Decentralized Environmental Notification Message).

The idea behind this work is to implement and test other parts of the ETSI protocol stack, focusing, in particular, on the ITS Transport and Networking layer. The two main protocols defined by ETSI for this layer are BTP (Basic Transport Protocol) and GeoNetworking. These protocols are designed to support a variety of heterogeneous application requirements and to provide communication in mobile environments without the need for a coordinating infrastructure, therefore, enabling Vehicle-to-Vehicle (V2V) as well as Vehicle-to-Infrastructure (V2I) communications.

The goal of this work is to further develop the MS-VAN3T simulation framework to investigate how these two protocols interact with the aforementioned Facilities layer and the different access layers supported by MS-VAN3T. This work thus increases and enhances the already powerful functionalities available in this framework.

With particular attention to the IEEE (Institute of Electrical and Electronics Engineers) 802.11p model present in MS-VAN3T, an analysis is made on how it is possible to simulate both V2V and V2I applications in an environment that does

not make use of the traditional transport and network layers (UDP or TCP over IP), but rather adopts the ETSI's BTP and GeoNetworking protocols to deliver packets among the nodes. These packets, sent in a simulation environment, can be easily dissected with a packet analyzer, such as Wireshark, so to extract meaningful information on the simulated network.

Although not all the features requested by ETSI are present in this work so far, the core functionalities for network management and packet handling necessary to fully support the Facilities layer services present in MS-VAN3T, have been implemented. We are going to see how we can make use of MS-VAN3T not only as a simulation tool but also as an emulator, being able to send packets into the real world with a physical interface.

Last but not least, with the aim of showcasing the efficiency and capabilities of MS-VAN3T for the development of new vehicular applications, a new application, based on V2I communications and called Traffic Manager, is presented. A detailed description of the application's logic, which fully relies on the BTP and GeoNetworking modules, together with the CA Basic Service, will be followed by an analysis of the obtained application performance results.

Acknowledgements

First of all, I would like to thank my thesis advisors Prof. Claudio Ettore Casetti, Marco Malinverno and Francesco Raviglione for giving me the chance to join this project and for the always kindly provided guidance, support and encouragement throughout the whole project.

To my mother, the one who from day one has always believed in me and has always given me her unconditional support, without her I could not have achieved any of this and to her I owe everything.

To my family and friends, thanks for always being there to make my life during my career more enjoyable and making this chapter of my life one that has served not only as professional growth but as personal growth as well.

Table of Contents

List of Figures

Acronyms

ITS Intelligent Transport Systems

VANET Vehicular Ad-Hoc Network

V2V Vehicle-to-Vehicle

V2I Vehicle-to-Infrastructure

V2X Vehicle-to-Everything

CA Cooperative Awareness

CAM Cooperative Awareness Message

DEN Decentralized Environmental Notification

DENM Decentralized Environmental Notification Message

ETSI European Telecommunications Standards Institute

IEEE Institute of Electrical and Electronics Engineers

BTP Basic Transport Protocol

GN GeoNetworking

RSU Roadside Unit

OBU On Board Unit

3GPP 3rd Generation Partnership Project

DSRC Dedicated Short Range Communications

C-V2X Cellular Vehicle-to-Everything

LLC Logical Link Control

MAC Medium Access Control

WAVE Wireless Access in Vehicular Environments

ITS-S ITS Station

ITS-FPDU ITS Facilities layer Protocol Data Unit

ITS-FSDU ITS Facilities layer Service Data Unit

ITS-FPCI ITS Facilities layer Protocol Control Information

BTP-PDU BTP Protocol Data Unit

T-PDU Transport Protocol Data Unit

BTP-SAP BTP Service Access Point

GN-SAP GN Service Access Point

MID MAC ID

DAD Duplicate Address Detection

LocT Location Table

LocTE Location Table Entry

EPV Ego Postiion Vector

LPV Long Position Vector

SHB Single-Hop Broadcast

GBC Geographically-Scoped Broadcast

DPD Duplicate Packet Detection

Chapter 1

Introduction

1.1 Intelligent Transport Systems

Almost from the birth of the automotive industry around the beginning of the 20th century, car manufacturers have made attempts to improve the safety systems of vehicles. Starting with the four-wheel hydraulic brakes, then with the introduction of built-in seat belts, later with the standardization of airbags and the list went on as the new technologies came along. For each of all these mentioned features, a fair number of years of prior investigation and testing went by before they were released and became a standard for the new products.

In the last 20 years, with this same goal of increasing safety capabilities for vehicles, significant efforts have been made towards the world of Intelligent Transportation Systems (ITS). This paradigm comes with a set of applications and services not only for road safety but for traffic efficiency, user experience and all kinds of different transportation needs as well. Both safety and efficiency types of application are entangled and benefit from each other and that is why vehicle industries, government entities and great number of academic researchers, are working together to standardize all the aspects of ITS systems.

The steps towards the realization of ITS systems have been fueled by major developments in Vehicular Ad-hoc NETworks (VANETs). This technology has been the most important enabler for ITS systems succeeding to provide fast and direct exchange of information needed for most ITS applications. The introduction of VANETs makes possible, with the use of Dedicated Short-Range Communication (DSRC), to have Vehicle-to-Vehicle (V2V) and Vehicle-to-Infrastructure (V2I) exchange of messages.

So far, one of the most explored ITS applications is the one related to Collision Avoidance, in which vehicles exchange messages with each other and, depending on the extracted information, each vehicle makes an autonomous decision. These

messages contain a set of useful information about the sending vehicle such as position, speed, acceleration and heading, and is organized and encoded in the so-called Cooperative Awareness Messages (CAMs) standardized by the European Telecommunications Standards Institute (ETSI) [1].

These Collision Avoidance applications benefits from the increasing amount of sensors and computing power equipped in modern vehicles, that until now has led to innovative features such as Autonomous Emergency Breaking. But these capabilities are limited to the local view of the vehicle, the idea is to increase this view by sharing information through VANETs with other vehicles and roadside units with the goal of enabling more complex and more effective applications. A big issue arises with the cooperative nature of these applications and the need of a big number of active players to have a detailed view of the scenarios and thus to have a good performance of the decision-making algorithms. This issue is addressed by [2], in which the authors analyze how the speed at which this new technologies are adopted and become widespread, modeled as percentage called Penetration Rate, is a big factor in the effectiveness of Collision Avoidance algorithms.

Taking into account other type of applications outside safety, the other big tranche of research attention in the world of ITS are the ones developed towards traffic efficiency. Traffic efficiency applications are not only designed to minimize the time that it takes to a vehicle to go from one point to another but one of the biggest motivation behind them comes from the reduction of fuel consumption and thus pollutant emissions. In particular, one interesting proposal is the application for Virtual Traffic Lights (VTL) that comes from the fact that, specially in big cities, a very low percentage of intersections are dictated by traffic lights which makes them very vulnerable to traffic loads. Furthermore the periodic and non-adaptive mechanism they make use of, result in high inefficiencies as well. The idea behind VTL is to enable vehicles to exchange messages with the information needed to coordinate them and simulate, as the name indicates, a virtual traffic light that could be seen by the drivers in the vehicle's built-in display. Different approaches have been taken to implement this application, one of them selecting periodically a vehicle to be the coordinator for each intersection that will broadcast V2V messages with the traffic lights signals. This approach is proven to be one of the most refined ones, since it enables a reduction traffic congestion up to 60% without the need of a coordinating infrastructure, i.e., all messages are V2V, at high vehicular densities [3][4].

Other applications thought for traffic efficiency include Platooning and applications for parking space search. Platooning is a quite old concept which consists in grouping vehicles together very close to one another and to behave as a single unit in order to increase the use of the road infrastructure. This application makes use of V2V exchange of messages not only to maintain the platoon but more importantly to manage the addition o subtraction of vehicle from the platoon

taking into account a number of details such as the speed and the location to which the group is headed. One of the use cases studied is the platooning of Heavy Duty Vehicles (HDVs) with the goal of taking advantage of the air drag reduction to reduce fuel consumption and therefore not only reduce emissions but also reduce fuel cost that for HDVs is a key factor [5]. On the other side of the spectrum of traffic efficiency applications we have the one towards parking space search that exploit V2I communications to allow roadside units (RSU) to create a view of the states of the parking spaces in its surroundings and suggest nearby vehicles in need of parking place the best option. The decision on the parking space suggested is made with the intention of reducing the time spent by the vehicle and also to reduce contention problems between them [6].

The enormous spectrum of these ITS applications and technologies that each day come to life, creates a necessity of protocols and standards to make possible the coexistence between all of them and that is why entities such us ETSI, IEEE (Institute of Electrical and Electronics Engineers) and 3GPP (3rd Generation Partnership Project) are crucial players in the world of ITS. In the next sections, a comparison is made of the different protocols and access technologies for connected vehicles with a major focus in the efforts made by ETSI in this regard.

1.2 VANETs

The success of ITS applications, as discussed, depends significantly on the communication between vehicles. As introduced in the previous section, VANETs have been the foundation over which vehicular communications have been constructed so far. VANETs were originally created as a variant of Mobile Ad-hoc NETworks (MANETs) to be suited for the different properties of their transmitting/receiving stations. The main characteristics of VANETs that differentiates it from it's counterpart, are the ability to cope with fast changes in the topology due to the elevated speed of vehicles, and to provide ultra low latency even with variable loads.

For communication to occur between vehicles and RoadSide Units (RSUs), vehicles must be equipped with some radio interface or OnBoard Unit (OBU) that enables short-range wireless ad hoc networks to be formed. Vehicles must also be fitted with hardware that permits detailed positioning information such as Global Positioning System (GPS) or a Differential Global Positioning System (DGPS) receiver. Fixed RSUs, which are connected to the backbone network, must be in place to facilitate communication and their distribution is a key factor depending on the application and protocol they are partaking [7].

We already introduced V2V and V2I as possible communication configurations for ITS applications and now for the sake of completeness, all communication configurations that are often referred to as Vehicle-to-Everything (V2X) are introduced

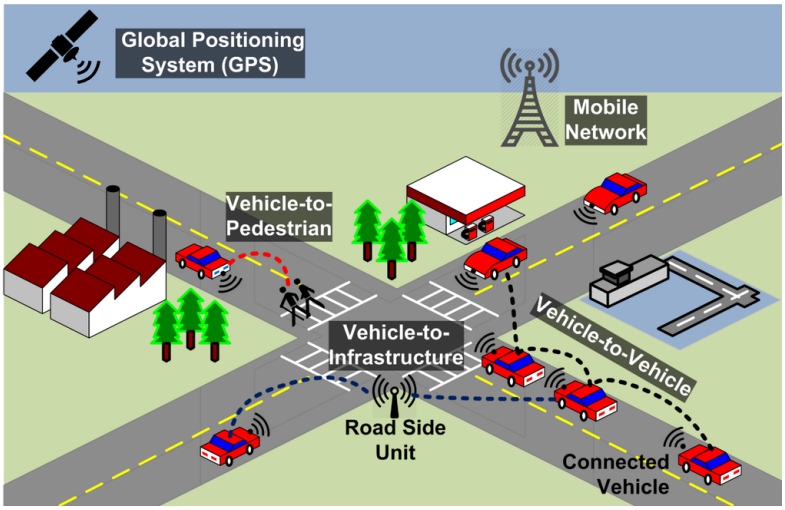


Figure 1.1: ITS V2X communications [8].

formally as following:

- Vehicle-to-Vehicle (V2V)
- Vehicle-to-Infrastructure (V2I)
- Vehicle-to-Network (V2N)
- Vehicle-to-Pedestrian (V2P)

In order to make the V2X communication paradigm possible, at the time of writing vehicular networks are mainly based on two families of access technologies:

- Dedicated Short Range Communications (DSRC) standards, mainly developed by ETSI and IEEE.
- Cellular based standards often referred to as C-V2X, mainly developed solely by 3GPP in recent years.

For the development of these communication technologies to thrive, from the beginning of this century the spectrum management organizations have made efforts to provide the vehicular communications with an exclusive band for ITS-based scopes. In 1999 the US Federal Communication Commission (FCC) allocated a 75 MHz bandwidth of the 5.9 GHz band to Dedicated Short-Range Communication (DSRC).The term “DEDICATED SHORT-RANGE COMMUNICATIONS” was used as a technology-neutral term for short-range wireless communication between vehicles and infrastructure [9]. After this, some follow up amendments where issued by the FCC. More precisely the band selected was from 5850 to 5925 MHz which

was divided into seven channels of 10 MHz each, recommended for ITS, and with a guard band of 5 MHz at the lower end as shown in Figure 1.2. Despite opposition from the automotive industry and entities devoted to the development of ITS, on November of 2020, the FCC has unanimously approved a First Report and Order reallocating a majority of the 5.9 GHz band away from connected vehicle technologies. More specially, it has repurposed the lower 45 megahertz at 5850-5895 MHz for unlicensed commercial uses such as Wi-Fi. This decision lets only the upper 30 megahertz of spectrum in the 5895-5925 GHz band to be still designated for intelligent transportation systems [10]. This decision constitutes a big setback for DSRC technologies and comes with a favored look towards C-V2X technologies.

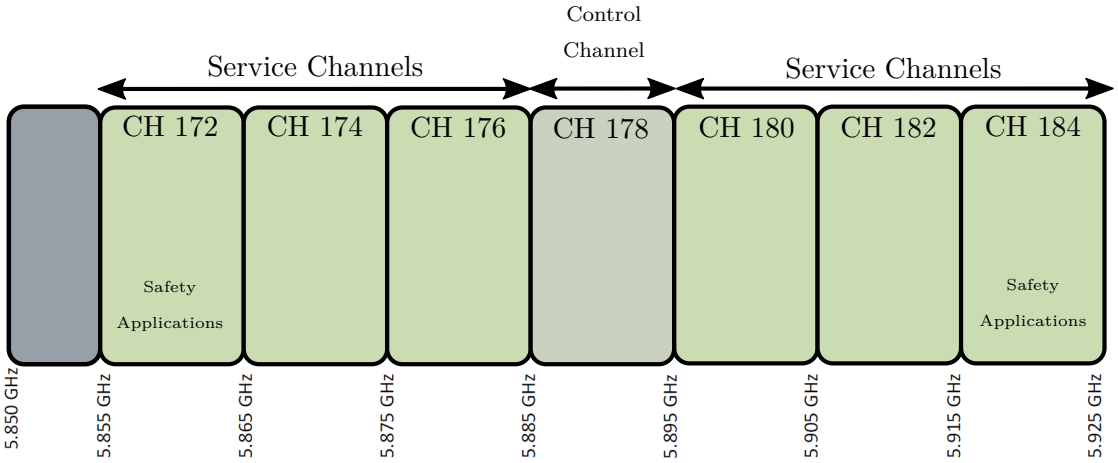


Figure 1.2: FCC spectrum allocation for the 5.9GHz band.

On the other hand, almost a decade later in Europe the European Commission with the Commission Decision 2008/671/EC [11], selected the band from 5875 to 5905 MHz to be used for ITS applications in the European Union. Later this band on harmonized by the European Conference of Postal and Telecommunications Administrations (CEPT) extending the band to go from 5855 to 5925 MHz [12]. The current spectrum allocation was designed following the technical recommendations TR 102 492-1 [13] and [14], where 5855–5875 MHz is assigned for non-safety related applications, 5875–5885 MHz for road safety and traffic-efficiency applications, 5885–5905 MHz for critical road-safety applications, and 5905–5925 MHz for road-safety and traffic-efficiency applications.

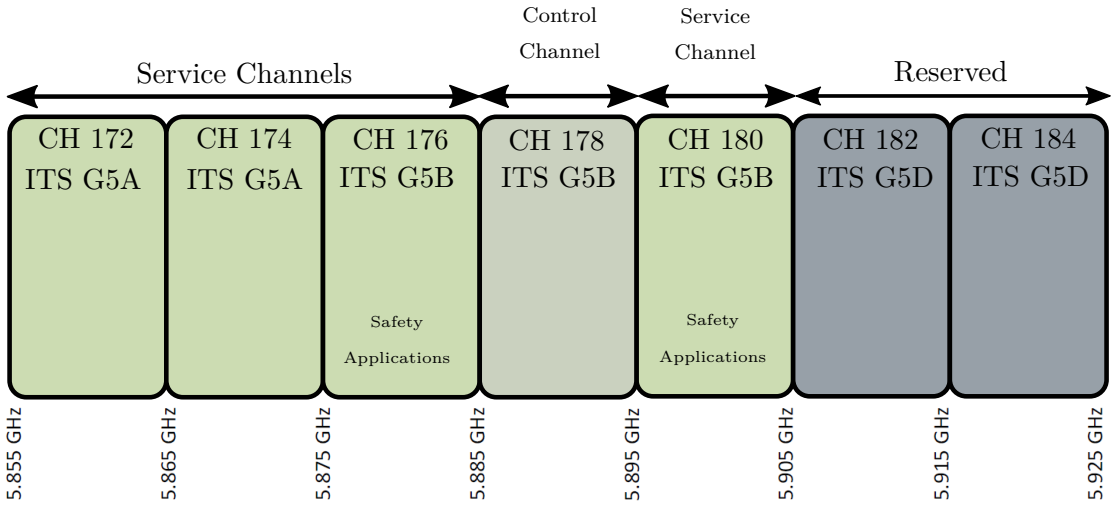


Figure 1.3: EU spectrum allocation for the 5.9 GHz band.

For the co-existence between DSRC and C-V2X technologies, the issue of the distribution of the ITS band for each technologies is of major importance. Different approaches must be carried on like the one suggested by [15] focusing, as this thesis, in the European scenario. In their work, the authors presented a possible solution comprised of three steps is presented where safety related ITS services from both technologies are ensured to work free of co-channel interference.

In the following section a deeper analysis of the different DSRC and C-V2X protocols is made to better understand the main limitations and opportunities offered by each of them and to have a detailed view of the vehicular networks landscape.

1.3 DSRC-based Protocols

1.3.1 WAVE

The IEEE WAVE (Wireless Access in Vehicular Environments) protocols is comprised by the IEEE 1609 family of standards [16][17][18][19], the IEEE 802.2 at the Logical Link Control layer (LLC) and the IEEE 802.11p at the MAC and Physical layer. This set of standards together form a communication stack especially designed to provide services for ITS enabling V2V, communication between OBUs, and V2I communications, between OBUs and RSUs.

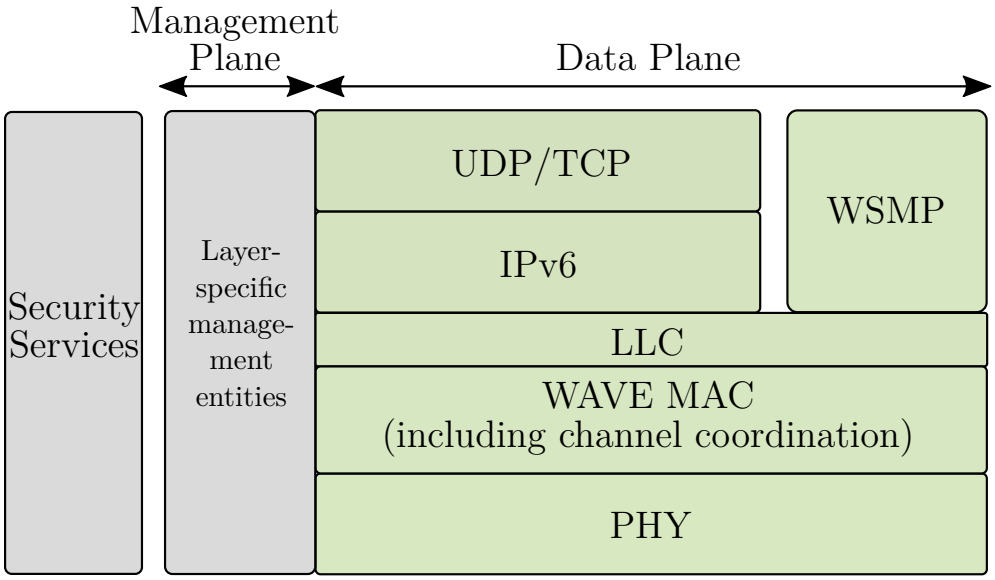


Figure 1.4: WAVE protocol stack [16].

These standards define how applications that utilize WAVE will function in the WAVE environment, based on the management activities defined in IEEE P1609.1, the security protocols defined in IEEE P1609.2, and the network-layer protocol defined in IEEE P1609.3. The IEEE 1609.4 resides above 802.11p and it supports the operation of higher layers without the need to deal with the physical channel access parameters allowing for multi-channel operation and provide for channel coordination between WAVE devices [7]. The higher layers, i.e, OSI Application, Presentation and Session layers, are not officially specified for WAVE.

The components of the WAVE protocol stack are illustrated in Figure 1.4. A data plane is defined for protocols carrying higher layer information, and a management plane is defined for management functions that indirectly support information transfer between layers.

Transport and Network Layers

IEEE 1609.3 specifies two data plane protocol stacks that simultaneously share a common lower stack at the data link and physical layers, the standard IPv6 and the WSMP (Wave Short Message Protocol) designed for optimized operation in a wireless vehicular environment.

The WSMP allows applications to directly control physical characteristics used in transmitting messages. The source application provides a PSID (Provider Service IDentifier) that uniquely identifies it and the MAC address of the destination

device, including the possibility of a group address. WSMs are delivered to the correct receiving entity based on the PSID. If the PSID value in a received message header represents a service that is not of interest to the WSM recipient, then the corresponding WSM data field is not processed.

As stated, the WAVE standards support IP version 6 (IPv6). IPv6 was selected over IPv4 due to the scalability issues associated with it. WAVE standards do not specify what transport and higher layer protocols may be used over IPv6. IP is appropriate for applications requiring the features provided by the IP suite and it's two popular transport protocols namely User Datagram Protocol (UDP) and Transmission Control Protocol (TCP) are supported.

Taking a peek at the Management Plane, in there the WAVE Management Entity (WME) services are associated with the various data plane entities to provide layer-specific functions necessary for a correct system operation. These functions include time synchronization for channel coordination and processing service requests from higher layers, generating and monitoring of WAVE Service Advertisements (WSAs). WSAs are generated, monitored, and, generally speaking, managed within the management plane by the WME. However, WSMP, which resides in the data plane, is used to transport WSAs in the way that is requested by the WME [16].

Data-Link layers

Moving down in the protocol stack, the LLC distinguishes between the two Transport and Network upper stacks previously presented by the EtherType field. EtherType is a 2-octet field in the LLC header, used to identify the networking protocol to be employed above the LLC sublayer. The EtherType field is specified in IEEE 802.2 but IEEE 1609.3 specifies it's usage in WAVE devices. The hexadecimal values indicating IPv6 and WSMP are 0x86DD and 0x88DC, respectively [16].

WAVE makes use of 802.11p standard for both MAC and Physical Layers as stated before but in order to further extended its functionalities, a MAC sublayer extensions to the IEEE 802.11p is specified in IEEE 1609.4 [19]. These extensions include channel coordination features in support of multi-channel operation.

802.11p

Lastly, as already introduced, the IEEE 802.11p is the standard used for the WAVE MAC and Physical Layers. This standard was designed to provide the specifications required to ensure interoperability between wireless devices attempting to communicate in potentially rapidly changing communications environments and in situations where transactions must be completed in very short time frames.

IEEE 802.11p is basically a modified version of IEEE 802.11a. On the Physical layer, 802.11p uses a reduced channel bandwidth of 10MHz in comparison with 20

MHz-wide band of the "a" amendment. Using smaller bandwidth allows to have a larger guard band between adjacent channels to avoid multi-path fading and doppler effect phenomena resulting in Inter-Symbol Interference [20]. As other Wi-Fi based technologies 802.11p utilizes OFDM (Orthogonal Frequency Division Multiplexing) based on 64 orthogonal subcarriers and it supports data rates from 3 Mb/s to 27 Mb/s, depending on the Modulation and Coding Scheme. The possible modulations are BPSK, QPSK, 16QAM and 64QAM.

On the MAC layer, 802.11p supports a new mode of operation in addition to ad hoc and infrastructure modes. This new mode of operation is called OCB (Outside the Context of a BSS). The idea is that vehicular nodes (STAs) operating in the OCB mode can send and receive messages without associating with a BSS in the conventional way. Additionally, 802.11e-based priority schemes are also considered for DSRC communication using the enhanced distributed channel access (EDCA). This mechanism defines four different Access Categories (AC) that divides the data into different priorities. From lowest priority to highest the ACs defined are [21]:

- AC_BK: Background.
- AC_BE: Best Effort.
- AC_VI: Video.
- AC_VO: Voice.

The objective is to allow service differentiation between safety and non-safety messages.

1.3.2 ITS-G5

Steering the attention now towards the European scenario of vehicular communications, since 2007 and taking the WAVE architecture as inspiration, ETSI has developed the ITS-G5 set of protocols. For the ITS Access Technologies ETSI uses IEEE 802.11p as the base standard as described in [22]. In spite of that, in more recent years ETSI has made the efforts to integrate new access technologies (such as LTE-V2X) to the stack offering guidelines for interconnection with higher layers, as outlined in [23].

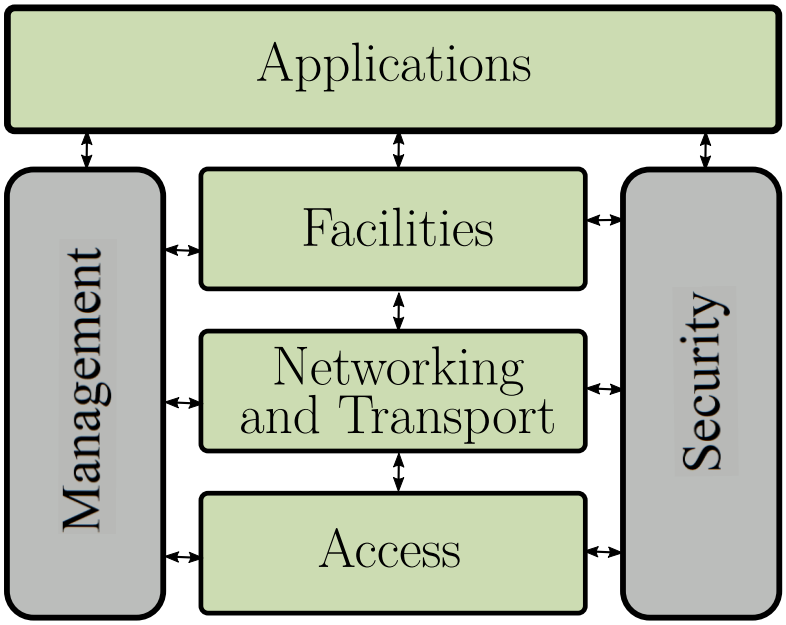


Figure 1.5: ITS station protocol stack.

The protocol stack of an ITS station specified in [24] attempts to follow the ISO/OSI reference model defined in ISO/IEC 7498-1. As shown in Figure 1.5 there are four data planes layer defined, namely:

- ITS Applications layer: defined in [25][26][27] which specify technical requirements for Road Hazard Signalling (RHS), Intersection Collision Risk Warning (ICRW) and Longitudinal Collision Risk Warning (LCRW) respectively.

- ITS Facilites Layer: defined in [28] and EN 302 637-2/3[29][30]. Here the Cooperative Awareness (CA) and Decentralized Environmental Notification (DEN) basic services are defined together with the format of the message managed by them for V2X communications.

- ITS Networking and Transport Layer : defined in [31][32] [33] [34] [35]. Here the GeoNetworking and Basic Transport Protocol are defined which correspond, respectively, to the layers 3 and 4 of the ISO/OSI reference stack.

- ITS Access Layer: defined in [22] where the outlines for the physical and MAC sub-layer are specified, and in [36] and [37] where the mechanism for Decentralized Congestion Control (DCC) is defined.

In addition, 2 vertical layers are defined :

- ITS management entity which is responsible for configuration of an ITS station, cross-layer information exchange among the different layers and others tasks [33].
- ITS security entity that provides security and privacy services, including secure messages at different layers of the communication stack, management of identities and security credentials, and aspects for secure platforms (firewalls, security gateway, tamper-proof hardware)[33].

In order to give context for the main focus of this work that revolves around the introduced ITS Networking and Transport Layer, a brief overview of the other layers of the stack is reported in the following sections.

Application and Facilites layers

With a more extended scope than WAVE, the ITS-G5 stack standardize the reference layers 5, 6 and 7 of the ISO/OSI stack (i.e. Session, Presentation and Application). For the Application layer, as already introduced, ETSI defines three main applications that rely on the exchange of CAMs (Cooperative Awareness Messages) and DENMs (Decentralized Environmental Notification Messages) for safety use cases. In [25] the RHS is introduced with the goal of increased awareness between ITS stations (ITS-S 3.1.2) and ITS stations with the driver for which two modes are described respectively. The originating mode considers the detection and signaling of a road hazard from an ITS-S to other ITS-Ss and the receiving mode considers the signaling of road hazards to the driver of the receiving vehicle when relevant to him. On the other hand, in [26] describes the ICRW application that is designed to detect potential collision risk between two or more vehicles or obstacles inside an intersection area. For this application several types of collision risk, with their potential use cases, are described such as: crossing collision, traffic sign violation, collision involving Vulnerable Road Users (VRU) and Rear end collision. Similarly, [26] the LCRW application is described with it's own types of collision risk such as: forward collision, forward/side collision and frontal collision.

Moving one step down the ITS-G5 stack, the Facilites is found providing support to ITS applications which can share generic functions and data according to their respective functional and operational requirements [24]. As described in [28] the facilities layer covers the 3 upper layer of the OSI reference model. The following three classifications of facilities, that then are organized as sub-layers, are defined:

- Application support facilities: Facilities that provide application support functionalities for the ITS Basic Set of Applications (ITS BSA). Examples of the application support facilities are the CAM and DENM management via the so-called CA (Cooperative Awareness) and DEN (Decentralized Environmental Notification) basic services.

- Information support facilities: Facilities that provide common data and database management functionalities for ITS BSA, such as the Local Dynamic Map (LDM).

- Communication support facilities: Facilities that provide services for communications and session management, such as the selection of the addressing mode for the V2X message transmission and provide the message dissemination requirements to the network and transport layer.

A brief description of the CA and DEN basic services has to be made since they provide the messages to be disseminated by the ITS Networking and Transport layer. The CA basic service is described in [29] as a facilities layer entity that operates the CAM protocol. It provides two main services: sending and receiving of CAMs. CAMs are generated periodically. The generation frequency is determined taking into account the change of own ITS-Ss status, e.g. change of position or speed as well as the radio channel load as determined by DCC in the ITS Access layer. Lastly, as described in the following chapters CAMs are transmitted in a single-hop manner. On the other hand the DEN basic service is described in [30] as an application support facility that constructs, manages and processes the DENMs. The construction of a DENM is triggered by an ITS-S application, not in a periodical way as the CA counterpart. A DENM contains information related to a road hazard or an abnormal traffic conditions, such as its type and its position. Also as described in the following chapters DENMs are typically disseminated to ITS-Ss that are located in a geographic area through direct vehicle-to-vehicle or vehicle-to-infrastructure communications.

Networking and Transport layers

The ITS networking and transport layer comprises protocols for data delivery among ITS stations and from ITS stations to other network nodes, such as network nodes in the core network (e.g., the Internet) [33]. Among the different possible networking modes this thesis focuses on the GeoNetworking as already introduced where an extensive description is made in Chapter 3. Is worth mention that support for IPv6, either with mobility support developed at IETF and ISO or over GeoNetworking, is provided. For what concerns the transport protocol, ITS-G5 defines the Basic Transport Protocol (described in Chapter 2), to be used on top of GeoNetworking. In the case of IPv6 networking modes, the already existing transport protocols UDP and TCP are to be used. Furthermore, IPv6 networking comprises methods to enable interoperability with legacy IPv4 systems [24].

Access layers

As introduced at the beginning of this section, for the ITS Access Technologies ETSI uses IEEE 802.11p (described in 1.3.1) as the base standard for the MAC and Physical layers, although later extended to support LTE-V2X [23]. On top of IEEE 802.11p, ITS-G5 defines one of the main features designed to cope with the highly volatile nature of VANETs, called Decentralized congestion control (DCC). As described in previous sections, one of the main challenges of VANETs comes from the fact that in given situations, there can be a high number of vehicles in communication range. The idea of DCC is to shape the data traffic injected by each communicating station to avoid channel congestion [36]. The ITS-S determines priorities between different messages and discards messages if application requirements exceed allotted resources.[37] describes the cross-layer operation of the DCC mechanisms for ITS-S while [36] describes the DCC mechanism from the access layer perspective.

1.4 Cellular-based Protocols

Ever since 3GPP completed the standardization of the Cellular Vehicle-to-Everything (C-V2X) technology in Release 14 [38], the interest in cellular based technologies for vehicular communications has skyrocketed. Despite being a recent technology compared to the stablished IEEE 802.11p with almost a decade separating their beginnings, the solution offered 3GPP brings very promising features to the table. This features include an increased radio coverage, the possibility to use the very well-stablished cellular infrastructure and a higher penetration rate due to the favored deployment for smartphones.

LTE-V2X

LTE-V2X was introduced in 3GPP's Release 14 which defined V2V communications based on D2D (Device-to-Device) communications defined as part of ProSe (Proximity Service) services in Release 12 and Release 13. A new channel in the Physical layer is introduced to allow the V2V communications called sidelink, which is especially designed for vehicular use cases, supporting nodes at high speeds and high density of nodes.

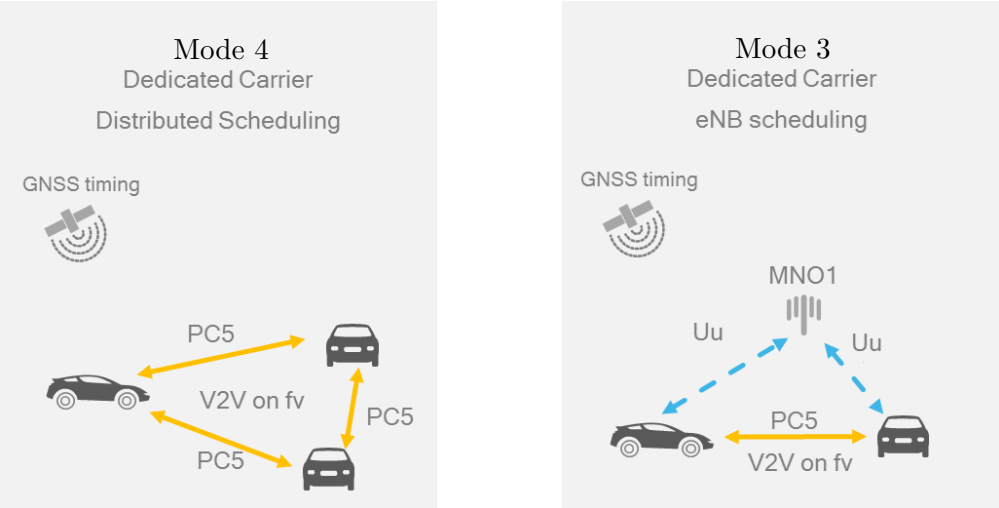


Figure 1.6: V2X communication over PC5 interface [38].

To enable V2X communications, 3GPP in [38] defines two modes of operation:

- V2X communication over PC5 interface: PC5 interface directly connects UEs (User Equipments) so that over-the-air V2X message from a UE is directly received by UEs around the transmitter. The V2X communication is supported using sidelink when the UE is inside LTE network coverage (a.k.a Mode 3), and also when the UE is out of network coverage (a.k.a Mode 4).

- V2X communication over LTE-Uu interface: LTE-Uu interface connects UEs with eNB (E-UTRAN NodeB) which plays the role of base station in the LTE networks. The UE may receive V2X messages (either unicast or broadcast) via downlink, while transmitting V2X messages via uplink. Differently from communications over PC5, communications over LTE-Uu are only supported when the UE is inside network coverage.

The target of Rel-14 work to support V2X service was mostly to provide data transport service for basic road safety service such as Cooperative Awareness Messages (CAM), Decentralized Environmental Notification Messages (DENM), so on [39]. In Release 15, 3GPP made the efforts to enhance the support for V2X scenarios offering new requirements for Vehicle Platooning, Advanced Driving, Extended Sensors and Remote Driving. Furthermore Release 15 introduces key functionalities such as support of Carrier Aggregation (CA) for transmission mode 4, support for 64-QAM, among others described in [39].

5G V2X

5G V2X is not going to replace LTE-V2X, but to supplement C-V2X in supporting those use cases that cannot be supported by LTE-V2X. NR V2X is designed to support V2X applications that have varying degrees of latency, reliability and throughput requirements. While some of these use cases require the transmission of periodic traffic, a large number of NR V2X use cases are based on reliable delivery of aperiodic messages. Furthermore, while some use cases require broadcast transmissions, others such as vehicle platooning are efficiently supported by the transmission of messages only to a specific sub-set of vehicles (UEs). 5G-V2X also introduces modifications to enhance the PC5 interface such as, modulation and coding scheme extended to 64-QAM, added possibility to use frequencies above 6 GHz, use of MIMO antennas and reduction of the TTI to 0.5 ms.

Chapter 2

Basic Transport Protocol

2.1 Introduction

The Basic Transport Protocol (BTP) is the Transport layer protocol defined by ETSI in [35] for ITS-G5. In a similar fashion to UDP, it provides an end-to-end, connection-less transport service for ITS ad hoc network. The transmission of packets is done in an unreliable manner, which means that there is no acknowledgement for reception and there is no reordering mechanisms provided.

The purpose of this protocol is to enable the ITS facilities layer to have access to the services of the GeoNetworking protocol by multiplexing the messages coming from both the CA basic service and DEN basic service. For what concerns the message reception, instead, BTP works de-multiplexing messages (i.e. CAMs and DENMs) from the GeoNetworking to the Facilities layer services.

To facilitate this multiplexing/de-multiplexing mechanism, as in UDP, BTP makes use of ports. Each port represents a communication endpoint that identifies the ITS Facilities layer service at the source or the destination.

The bridge over which the messages are exchanged between the two layers (i.e. BTP and Facilities) is called BTP-SAP (BTP Service Access Point). Through it the ITS-FPDU (ITS Facilities layer Protocol Data Unit) together with the ITS-FPCI (ITS Facilities layer Protocol Control Information) travel encapsulated inside service primitives with all the parameters necessary for the Facilities layer to fully exploit the services provided by GeoNetworking. Most of the ITS-FPCI is meant for GN and is then encapsulated with the BTP-PDU (BTP Protocol Data Unit) into the GN service primitives, reviewed in the next chapter, to be sent over the GN-SAP (GeoNetworking Service Access Point).

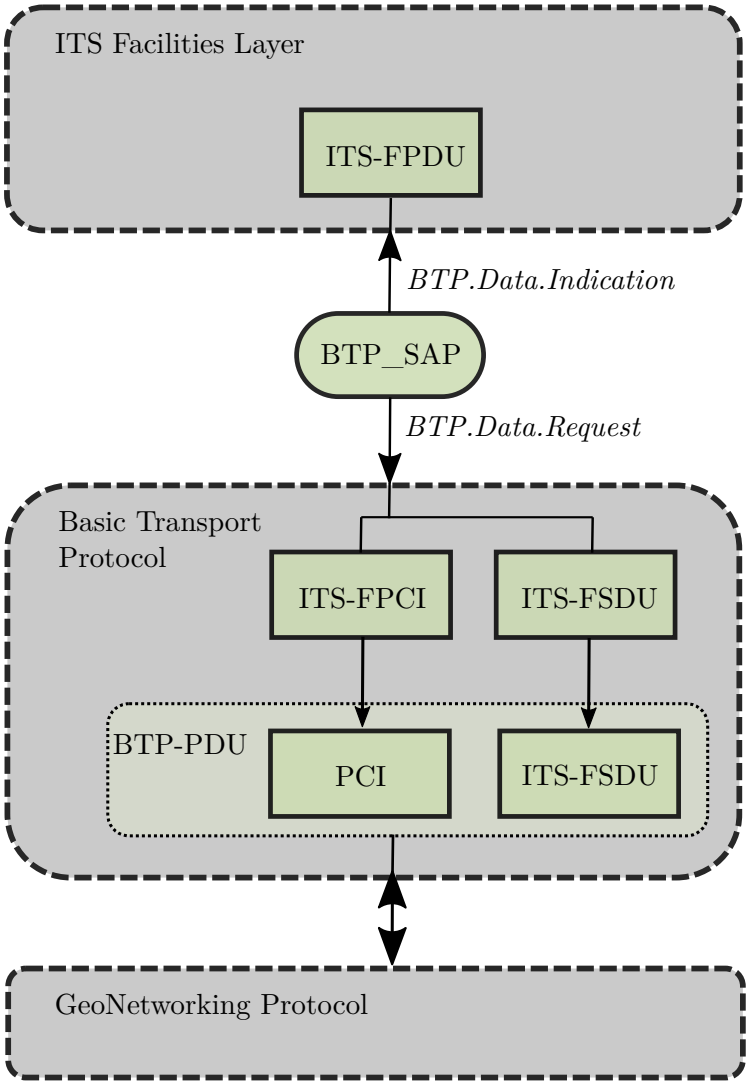


Figure 2.1: SAPs, SDUs and PDUs relevant for the BTP.

BTP is designed to be a lightweight protocol and to need minimal processing requirements. Its goal is to allow a fluid and harmonious communication between the Facilities layer and the GeoNetworking by acting almost as a mere bridge for the control information exchange and add a very small header to specify to which facilities layer service the message belongs to.

As can be seen in the Figure 2.1 the two service primitives for message exchange are the *BTP-Data.Request* used by the Facilities layer to request sending a BTP-PDU and the *BTP-Data.Indication* used by the BTP to indicate a reception of an ITS-FSDU to the Facilities layer.

2.2 BTP packet structure

As specified by ETSI in [35], a BTP packet is encapsulated in a frame comprising :

- The MAC header, namely the header of the MAC protocol of the ITS access technology.
- The LLC header, namely the header of 802.2 LLC/SNAP specified in ISO/IEC 8802-2.
- The GeoNetworking header, namely the header of the GeoNetworking packet where an optional security header is defined in [34] and a extended for media-dependent GeoNetworking functionalities. This header will be further studied in the next chapter of this thesis.
- The BTP header, namely the header of the Basic Transport Protocol.
- The payload represents the user data that is created by upper protocol entities, i.e. the ITS-FSDU, and passed to the BTP entity for transmission.

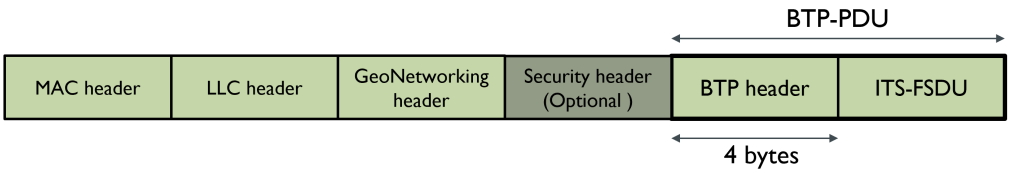


Figure 2.2: BTP-PDU encapsulated within the full ITS-G5 frame.

The BTP header combined with the payload is defined as the BTP-PDU as seen in the Figure 2.2 that is then treated as payload for the GN service primitives.

2.3 BTP header

The BTP header is a 4-byte protocol header comprised of two 16 bit integer fields. The meaning of these fields are dictated by the type of the BTP packet, which depends on the need for interaction between the two ends of the communication session. ETSI defines two types of packets:

- BTP-A: for interactive communication session.
- BTP-B: for non-interactive communication session.

2.3.1 BTP-A

The BTP-A header, since it is meant for interactive packet transport, which implies that a reply might be issued to the source of the message, it is comprised by a destination port and a source port. The destination port identifies the Facilities layer entity to which the BTP-PDU is intended to, while the source port identifies the Facilities layer entity that has issued the message and to which address a reply for said BTP-PDU.

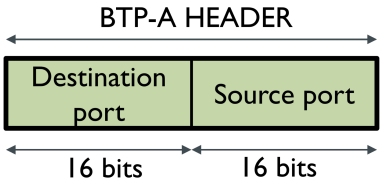


Figure 2.3: BTP-A header.

2.3.2 BTP-B

The BTP-B header unlike BTP-A is meant for non-interactive packet transport and thus does not carry the source port. It carries only the destination port to which the BTP-PDU is intended to and the destination port information. This last field has a default setting value of 0.

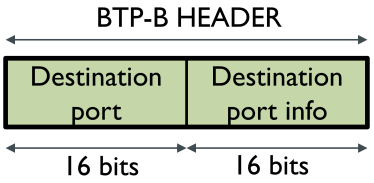


Figure 2.4: BTP-B header.

Both CA and DEN basic services from the Facilities layer, which are implemented in MS-VAN3T, are non-interactive services given that after a reception of a message of either service the receiver does not have to reply. For this reason they will always make use of BTP-B for the transport of their ITS-FSDU.

Lastly, as in the IP protocol suite, BTP adopts the concept of "well-known ports" that defines fixed port numbers to each specific Facilities layer entities and related applications. These well known ports are defined in [40] together with their respective destination port information.

Chapter 3

GeoNetworking protocol

3.1 Introduction

As stated in [31], GeoNetworking (GN) is a network-layer protocol for mobile ad hoc communication based on wireless technologies, such as ITS-G5. This protocol is characterized by, as the name implies, the use of geographical positions for the dissemination of information and transport of data packets. Putting aside the different applications for which this protocol was originally meant, the goal of GN is to provide both Vehicle-to-Vehicle (V2V) and Vehicle-to-Infrastructure (V2I) communication for Vehicular Ad-hoc NETworks (VANETs).

GeoNetworking is designed to cope with the volatile nature of VANETs in terms of mobility and topology, working in a connectionless and fully distributed fashion. The idea of this protocol is to support the wide spectrum of applications defined for ITS systems offering key features for each of them. Focusing on the applications analyzed in this work, for road safety it enables periodic transmission of status messages at high rate and for traffic efficiency it enables multi-hop dissemination of packets in geographical regions for emergency warnings.

GeoNetworking exploits the geographical position of nodes to allow packets be addressed to a node by its position or even to multiple nodes in a specific geographical region. For this to work GN assumes that every node has a partial view of the network topology. In this way whenever a node receives a packet, it process the geographical position in the packet's address resulting in a forwarding decision depending on the information it has on the topology.

3.1.1 Communication scenarios

The geographical addressing and forwarding capabilities provided by GN allow the definition of a set of communication scenarios with different supported forwarding

schemes. Each scenario is suited for different ITS applications and they may be combined with each other if needed.

The set of communication scenarios, defined in [32], for which GN is designed to support are:

- Point-to-Point: Communication starts at a single ITS station and ends at one ITS station. This scenario is identified by the *GeoUnicast* forwarding scheme featured in GN. In this scheme when a node sends a unicast packet, it first determines the destination node position and then forwards the data packet to a node towards the destination, which in turn re-forwards the packet along the path until the packet reaches the destination. In Figure 3.1 we can see a Point-to-Point scenario for V2V where before reaching the destination a packet is forwarded 3 times by intermediate nodes.

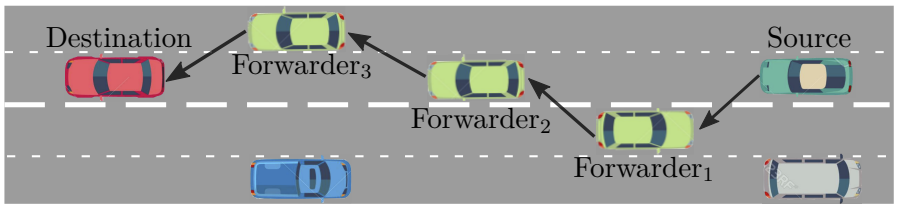


Figure 3.1: Point-to-Point.

- Point-to-Multipoint: Communication starts at a single ITS station and ends at multiple ITS stations. This scenario is identified by the *Topologically-scoped broadcast* forwarding scheme featured in GN. In this case a node broadcast a packet to all the neighbor nodes, i.e. all nodes in communication range, then each neighbor re-broadcast the packet to it's neighbors and so on (Figure 3.2). Indeed the ITS CA service messages make use of this forwarding scheme, more specifically of the so called *Single-hop broadcast* in which messages are not re-forwarded by the receivers in the one-hop neighborhood.

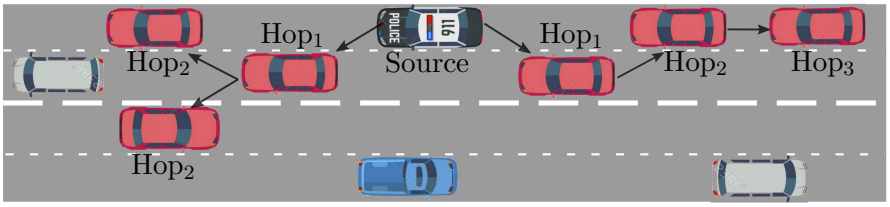


Figure 3.2: Point-to-Multipoint.

- GeoAnycast: Communication starts from a single ITS station and ends at an

arbitrary vehicle ITS station within a geographical area. In this scenario packets are forwarded hop-by-hop until they reach the destination area determined by the packets. Figure 3.3 illustrates a GeoAnycast scenario where a packet is re-forwarded 2 times until it reaches the first node inside the destination area specified by the source.

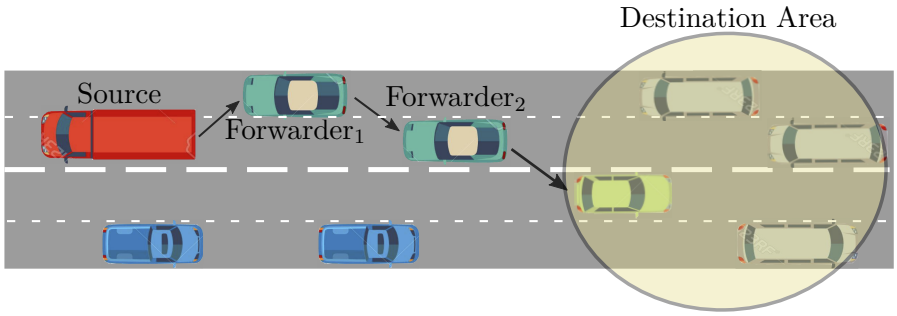


Figure 3.3: GeoAnycast

- GeoBroadcast: Communication starts from a single vehicle ITS station and ends at multiple vehicle ITS stations within a geographical area. This scenario uses the same forwarding scheme as in GeoAnycast with the difference that when the packet reaches a node in the destination area, all the nodes inside it rebroadcast the packet (Figure 3.4). This forwarding scheme is the one used for the ITS DEN service messages.

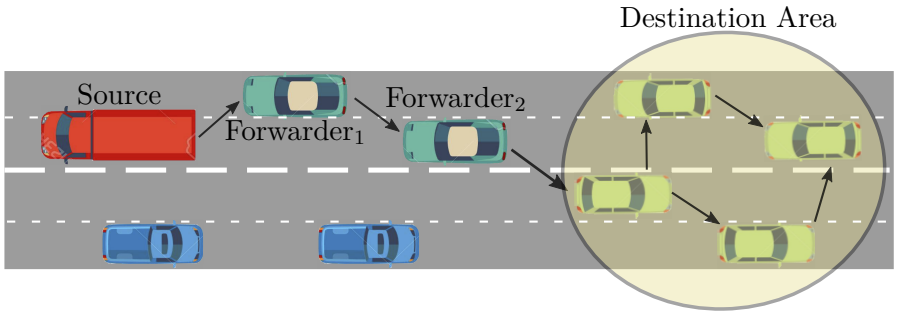


Figure 3.4: GeoBroadcast.

3.1.2 Network architecture

Before barging in the technical specifications of GN, the network architecture for ITS has to be analyzed. The central component of the architecture is the ITS station which has two main roles. The first role is to be a network node acting as source, forwarder or sink. The second role is, by being placed at the edge of

the ITS network, to connect the different networks via an ITS station internal network as seen in Figure 3.5. ETSI envisions in [33] for ITS stations to be able to communicate via:

- an ITS ad-hoc network
- an ITS access network
- a public access network
- a private access network
- one of the access networks into the core network

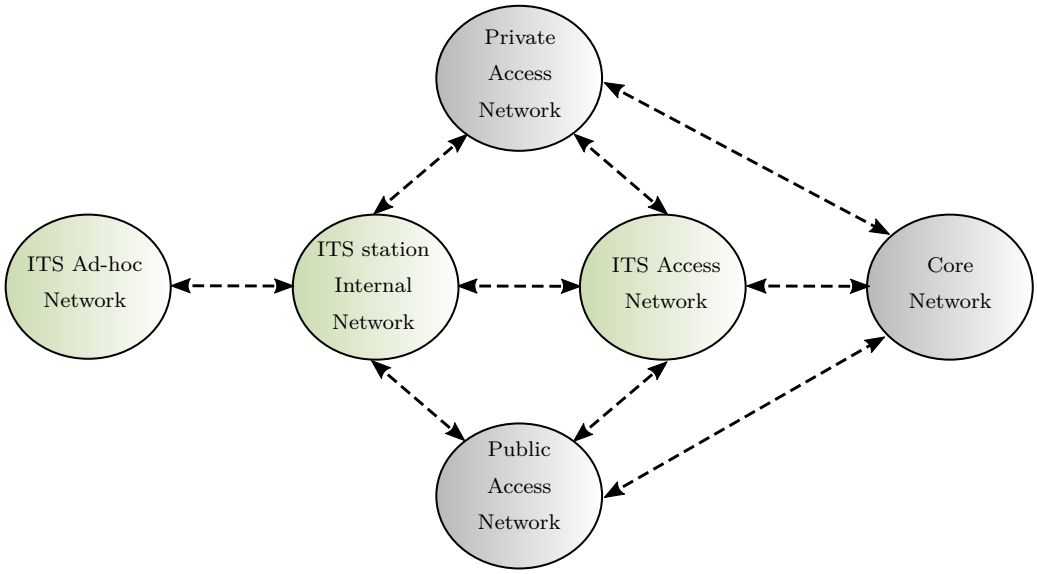


Figure 3.5: GN network architecture.

Being the main component of the network architecture, the following types of ITS stations are identified:

- vehicle ITS station
- personal ITS station
- roadside ITS station
- central ITS station

At the same time vehicle ITS stations and roadside ITS stations consist of two types of sub-components, the Communication and Control Unit (CCU) and the Application Unit (AU). In general, each CCU executes a communication protocol stack while each AU runs a single or a set of applications which make use of the CCU's communication capabilities. The CCU shall be equipped with at least a single ITS external communication interface to provide connectivity to the ITS ad hoc network or the different aforementioned access networks. On the other hand the CCUs and the AUs can be equipped with one or multiple ITS internal communication interfaces to connect with each other and for AUs to obtain connectivity to the networks via the external communication interface of the CCU.

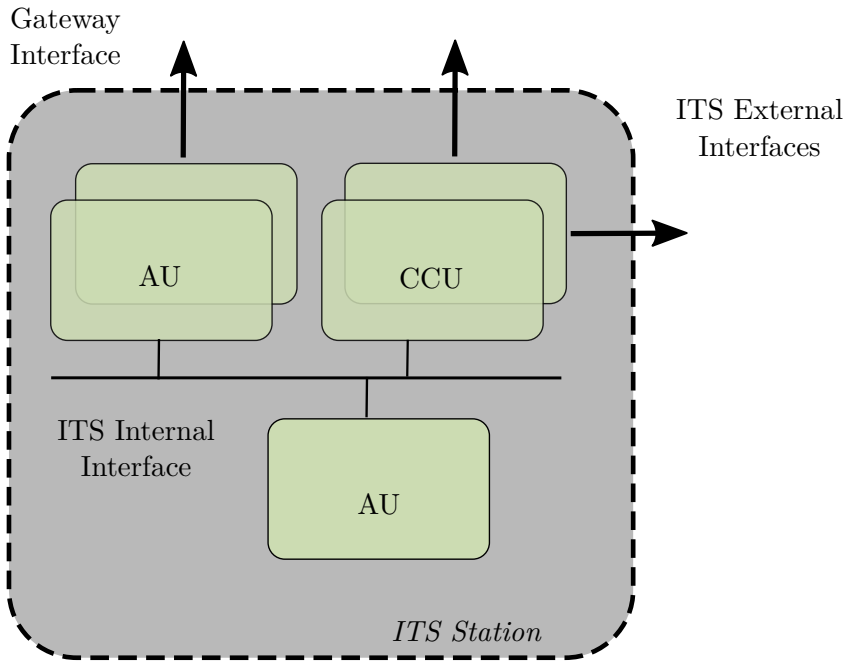


Figure 3.6: ITS station internal architecture.

Lastly, the CCU can be further sub-divided into logical network components for each of the different network layers supported by an ITS station:

- Ad-hoc router: this network component is the main focus of this work. Is the one that operates with the ITS ad-hoc network and executes the GeoNetworking protocol.
- Mobile router: provides IP connectivity of the ITS station internal network to an access router.

- Access router: being part of the ITS access network, it offers IP connectivity to ITS stations.

- Access network gateway: connects an access network to the core network.

3.1.3 GeoNetworking in the ITS station protocol stack

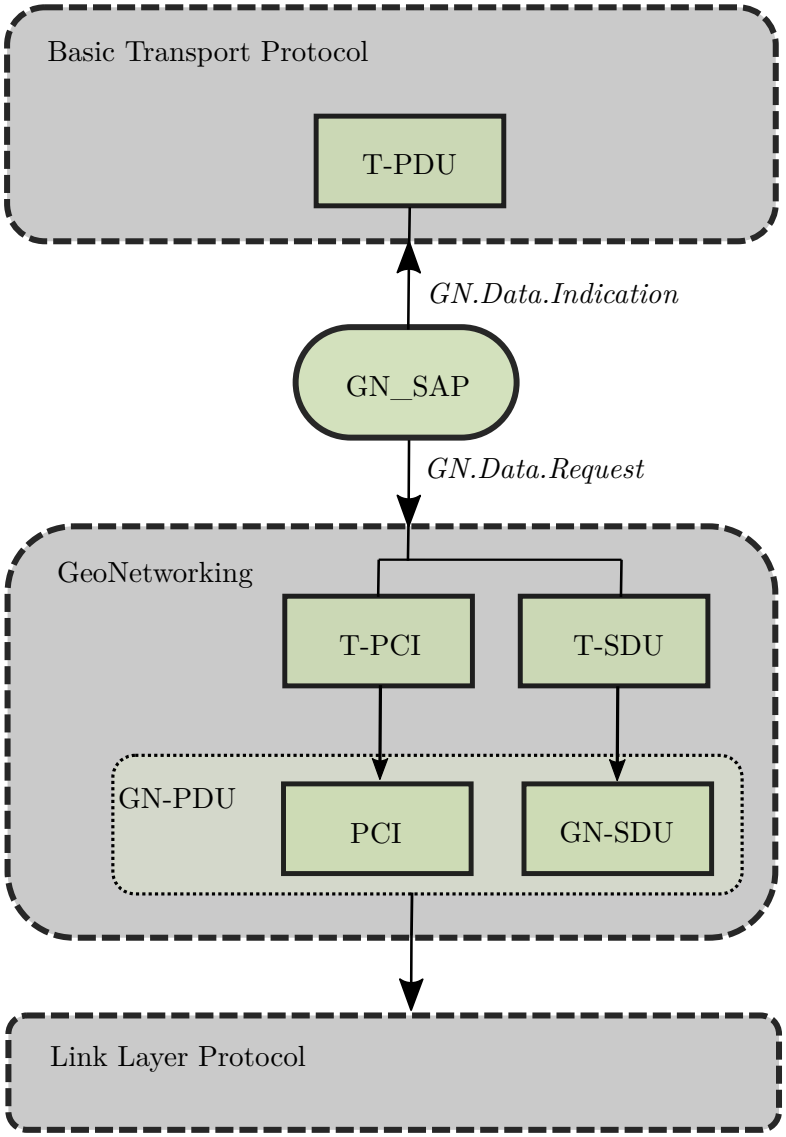


Figure 3.7: SAPs, SDUs and PDUs relevant for the GeoNetworking protocol.

As stated the GeoNetworking protocol is executed by the Ad-hoc router, more specifically the GeoAdhoc router, providing the transport of packets in the Ad-hoc Network. GeoNetworking provides services to the ITS Transport Protocol, such as the Basic Transport Protocol (BTP) as specified in [35], and the GeoNetworking to IPv6 Adaptation Sub-Layer (GN6ASL) as specified in [41]. The services are provided via the GeoNetworking Service Access Point (GN_SAP) using service primitives that encapsulate different parameters, many of them for or from the ITS Facilites layer, and the PDU of the upper protocol entity, i.e. T-PDU or GN6-PDU. The focus of this work is on the implementation of GeoNetworking in conjunction with BTP, thus from here on only BTP as upper protocol is considered. The T-PDU from BTP is considered in GN as a Service Data Unit (SDU), i.e. payload. The SDU is combined with the Protocol Control Information (PCI) available in the service primitives to create the GN-PDU that is later transmitted to the ITS Access layer. In the opposite side of the communication intuitively the PCI is extracted from the GN-PDU and together with the SDU is sent to BTP via the GN_SAP.

3.2 GeoNetworking address

The GeoNetworking address is the address used in the header of a GN packet to identify the GN entities communicating in the Ad-hoc Network. The GN address must be unique. As seen in the Figure 3.8 the first bit of the GN address is reserved to identify if the address has been configured manually. The ST (ITS-**S**tation **T**ype) field is a 5 bit field used to identify the ITS station type as specified by [42]. The MID (**M**AC **ID**) field corresponds to the access layer address which in the case of ITS-G5 MAC layer, as specified in [43], the 48-bit MAC layer address is used.

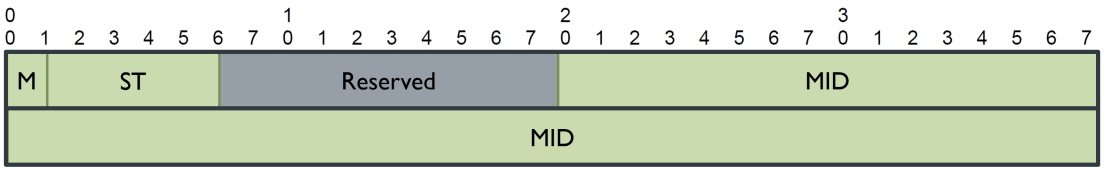


Figure 3.8: GeoNetworking Address.

The GN address can be configured in three different ways [34]:

- Auto-address configuration: where the MID field used is arbitrarily assigned by the GeoAdhoc router, for example with a random address generation.
- Managed address configuration: here the MID field is provided by the ITS Networking and Transport Layer Management entity which if needed can request for an address update.

- Anonymous address configuration: here the MID is provided and controlled by the security entity which executes all the mechanisms specified in ETSI TS 102 723-8 [44].

Given the possible existance of duplicate addresses mostly due to the Auto-address configuration, every GeoAdhoc router executes the *duplicate address detection* where on packet reception if indeed a duplicate address is detected, a new one must be requested to the ITS Networking and Transport Layer Management entity.

3.3 Data structures

Every GeoAdhoc router in addition to the GN address have a set of data structures which are the main instruments for enabling geographical addressing and forwarding. These data structures are the following [34]:

- Location Table (LocT)
- Ego Position Vector
- Sequence Number
- Location Service packet buffer
- Forwarding packet buffer

In this thesis the forwarding capabilities and therefore the GeoUnicast forwarding scheme of GN have not been implemented yet and are left as one of main features to be developed in future work. For this reason both Location Service packet buffer and Forwarding packet buffer are not further analyzed.

3.3.1 Location Table

This data structure is used by the GeoAdhoc router to store information about the other ITS stations (ITS-S) that are (or were) located in the neighbourhood and in this way sustain a view of the topology of the Ad-hoc Network. For every ITS-S a Location Table Entry (LocTE) exists which contains the following data elements [34]:

- GN address of the ITS-S
- Link Layer address of the ITS-S
- Type of the ITS-S
- Version of the GeoNetworking protocol of the ITS-S

- Long Position Vector (LPV) of the ITS-S (3.4.4)
- Flag called *LS_PENDING* which indicates that a Location Service (LS) is in progress.
- Flag called *IS_NEIGHBOUR* which indicates if the GeoAdhoc router of the ITS-S is in direct communication range.
- Duplicate packet list
- The timestamp of the last packet received
- Packet data rate PDR

Every LocTE has a lifetime attached to it, and is automatically removed from the LocT when that lifetime expires.

3.3.2 Ego position vector

This data structure is used by the GeoAdhoc router to store information about itself which is periodically updated each second by default. The Ego Position Vector (EPV) contains the following data elements [34]:

- Geographical position (expressed in Longitude and Latitude)
- Speed
- Heading
- Timestamp of the last EPV update
- Accuracy of the geographical position

3.3.3 Sequence number

This data structure is used to store the sequence number to be used in the next GN packet to be sent which is only used by packets need to re-forwarded, i.e, multi-hop packets only.

3.4 GeoNetworking packet structure

A GeoNetworking packet is part of the overall frame structure showed in figure 3.9:

- The MAC header, namely the header of the MAC protocol of the ITS access technology. The MAC protocol may add additional protocol elements, such as a trailer for the MAC FCS as in ITS-G5 [43].

- The LLC header, namely the header of 802.2 LLC/SNAP specified in ISO/IEC 8802-2 with the Ethernet Type field 0x8947 indicating GeoNetworking as the LLC transport protocol.
- The GeoNetworking header, namely the header of the GeoNetworking packet.
- The optional payload represents the user data that are created by upper protocol entities, i.e., the T-PDU.

Figure 3.9: GeoNetworking packet struture.

3.4.1 GeoNetworking header structure

The GeoNetworking header is comprised of a Basic Header(3.11), Common Header(3.12) and an optional Extended Header.

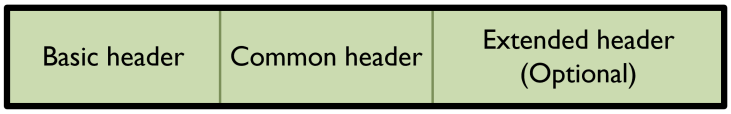


Figure 3.10: GeoNetworking Header.

Packet header types

The following GeoNetworking packet types are defined, each one with an specific extended header [34]:

- GUC (GeoUnicast) packet.
- TSB (Topologically-Scoped Broadcast) packet.
- SHB (Single-Hop Broadcast) packet (3.15).
- GBC (GeoBroadcast) and GAC (GeoAnycast) packet both with the same extended header structure (3.16).
- BEACON packet header (3.14).

- LS Request and LS Reply packet headers.

As hinted in previous sections, the SHB packet is the one used by GN to transport messages from the ITS CA entity and on the other hand, the GBC packet is the one used for messages from the ITS DEN entity. These two types of GN packets together with the Beacon packet are the focus of this work and are the ones for which a further analysis will be made.

3.4.2 Basic header

The first header in every GN packet is the Basic Header. The first 4 bits denote the version of the GN protocol and the 4 following bits denote if the next header is the Common header, as shown in Figure 3.10, or the optional *Secured packet*. The lifetime expresses the amount of time the packet is allow to spend buffered and lastly the Remaining Hop Limit indicates the amount of hops left the packet is allow to make (gets decreased each hop).

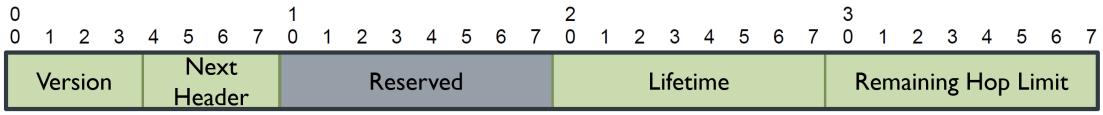


Figure 3.11: Basic Header.

3.4.3 Common header

Following the structure in Figure 3.10 we have the Common header. Here the Next Header field indicates the upper protocol to handle the current packet (which, in the case of CAM and DENM, will always be BTP-B). The Header type and sub-type fields denote which extended header follows this Common header, i.e the type of this GN packet. The traffic class is parameter that is passed through the service primitives and that comes from, and is destined to the ITS Facilities entity. A flag is used to indicate if the GeoAdhoc router belongs to mobile ITS-S.

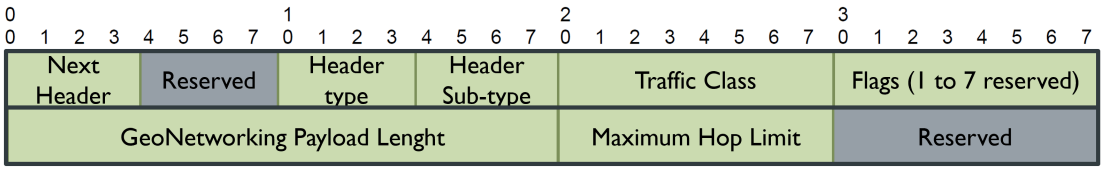


Figure 3.12: Common Header.

The GeoNetworking payload lenght is the lenght of the T-PDU carried by this GN packet. Lastly the Maximum Hop Limit, unlike the Remaining Hop Limit from the Basic header, is not decreased each hop.

3.4.4 Position vectors

Before proceeding with the structure of either of the extended headers, we describe here the Position Vectors. This structure is one of the major enablers of the GN protocol being responsible of all the geographical capabilites featured in the protocol. There are two types of position vectors :

- Long Position Vector (LPV)
- Short Position Vector (SPV)

Both structures are identical with the only difference being that SPV lacks the last 4 bytes of the LPV shown in Figure 3.13, i.e., Position Accuracy Indication (PAI), speed and heading. These structures act as the IP address in the IP protocol, with the LPV used in all GN packets used as Source address and the SPV used as Destination address for Point-to-Point type GN packets.

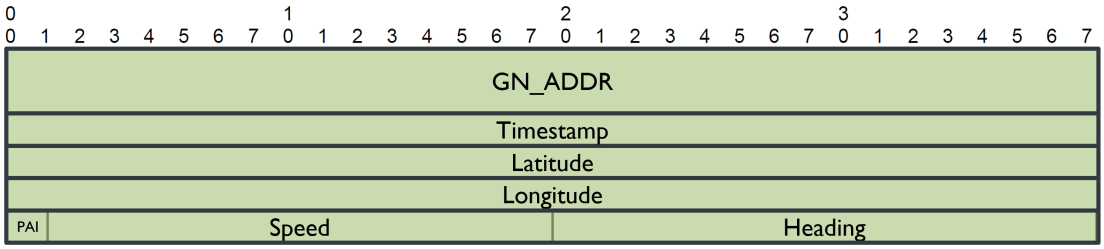


Figure 3.13: Long Position Vector.

Focusing on the LPV, the GN_ADDR is set with the GeoAdhoc router's own GNAddress configured at start-up and all the other fields are set with the values of the EPV (Section 3.3.2).

3.4.5 Extended headers

To conclude the analysis of the GN packet structure the last header to be analyzed is the Extended header. As discussed, each type of GN packet (Figure 3.4.1) has a different Extended Header.

Beacon

The beacon packet is used to periodically advertise a GeoAdhoc router's position vector to its neighbours. The idea is not to only use this packet for all GeoAdhoc routers in range to add new entry in the Location Table for a better topology view, but also to be sent periodically so that this information get updated in the aforementioned Location Tables, and thus be more meaningful. The Beacon packet is only meant for the ITS Network and Transport layer and therefore does not have a T-PDU and its extended header only consist of the source's LPV.

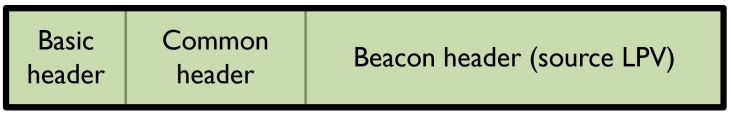


Figure 3.14: Beacon packet.

SHB

The Single-Hop Broadcast (SHB) packet is used by GN to transport messages from the ITS CA service entity which are intended to be sent to all neighbours of the GeoAdhoc router. The SHB extended header is comprised of the source's LPV and an optional field for media-dependent applications defined in [45]. For CAM messages this optional field is set to zero. The SHB packet is completed with the T-PDU as shown in 3.15 and sent to the ITS Access Layer.

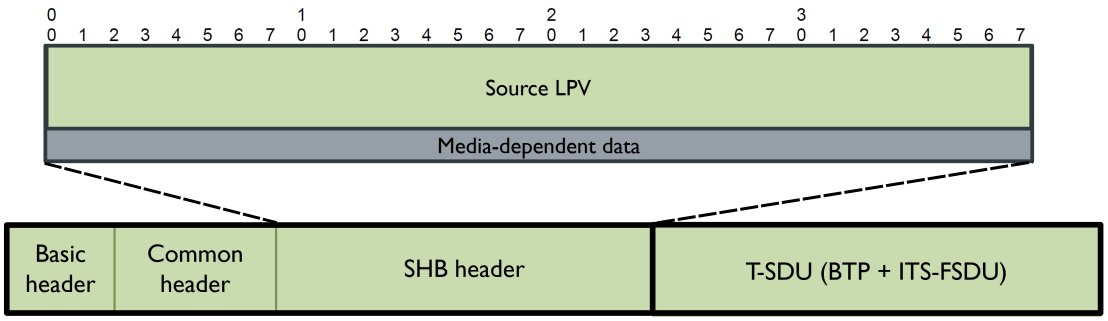


Figure 3.15: Single-hop broadcast packet.

GBC

The GeoBroadcast (GBC) packet is used by GN to transport messages from the ITS DEN service entity which are intended to be sent to all GeoAdhoc routers

within Geographical Area (GeoArea). The GBC extended header is comprised of the GeoAdhoc router's Sequence Number data structure (3.3.3), LPV and the GeoArea to which the packet is intended to. The SHB packet is completed with the T-PDU as shown in 3.16 and sent to the ITS Access Layer.

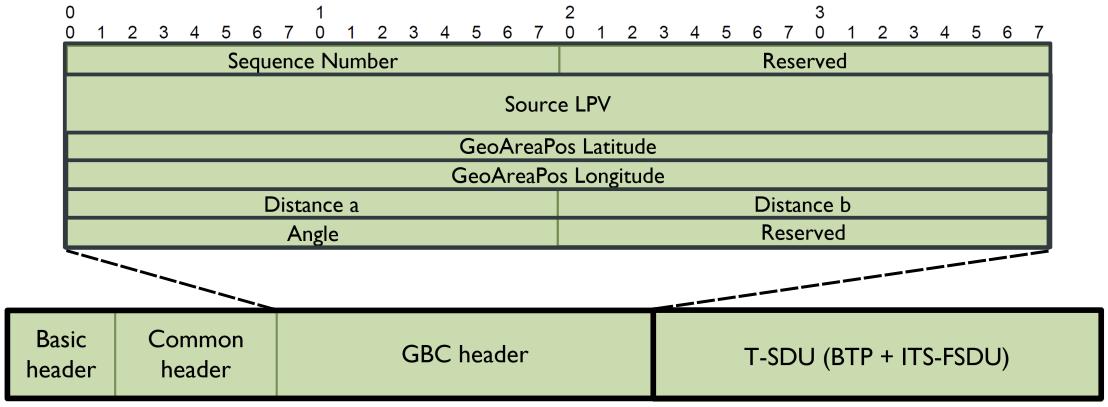


Figure 3.16: GeoBroadcast packet.

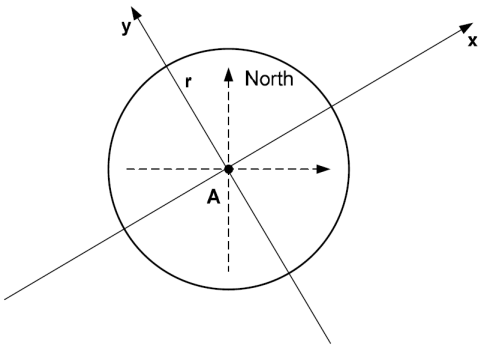
The geographical area fields of the GBC header may define 3 different shapes as specified in [46]:

- circular area, shown in Figure 3.17a.
- rectangular area, shown in Figure 3.17b.
- ellipsoidal area, shown in Figure 3.17c.

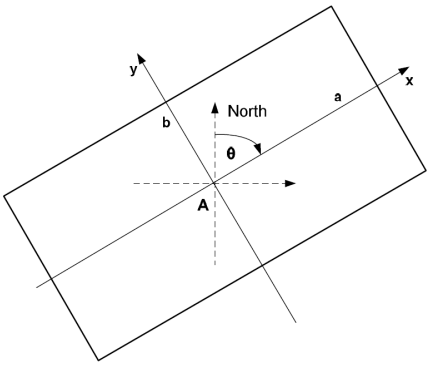
For the definition of the geographical area, 5 fields are found in the GBC header which are:

- Longitude
- Latitude
- Distance a
- Distance b
- Angle

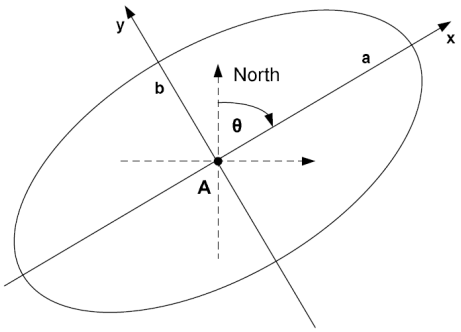
The longitude and latitude define the center of the shape. For a circular area, `distance a` defines its radius while `distance b` and `angle` variables are always set to zero.



(a) Circular Area



(b) Rectangular Area



(c) Ellipsoidal Area

Figure 3.17: Possible geometric shapes for the geographical area.

For a rectangular area, `distance a` defines the distance between the center point and the short side of the rectangle. Then,`distance b` defines the distance

between the center point and the long side of the rectangle. Lastly, the `angle` field defines the so-called *azimuth* angle which describes the rotation of the long side of a rectangle from the north axis (i.e., θ angle shown in Figure 3.17b).

For a ellipsoidal area, `distance a` defines the length of the long semi-axis, while `distance b` defines the length of the short semi-axis. Similarly to the rectangular area, the azimuth `angle` field defines the rotation of the long semi-axis of the ellipse (i.e., θ angle shown in Figure 3.17c).

Chapter 4

MS-VAN3T framework

4.1 Introduction

Prior to the actual implementation of new features involving innovative technologies, the automotive industry has always invested a great amount of time in the application testing phase. It has been already mentioned that the penetration rate, i.e., the percentage of nodes equipped with communication capabilities in the vehicular network, is a determinant factor for the success of most of the ITS applications. Because of the cooperative nature of ITS applications, normally huge fleets of connected vehicles are needed for the testing phase, and this brings a lot of constraints, specially in the economic and logistic domains. That is why the modeling and simulation of realistic vehicular communications environments is of paramount importance for the development and testing of new ITS applications. One of the main challenges of testing vehicular protocols in simulated vehicular scenarios, is the need for bidirectional coupling of both network and road traffic simulations.

This thesis puts its basis on a open-source vehicular network simulation framework built in the ns-3 simulator and that makes use of the SUMO (Simulation of Urban MObility) simulator, taking care of the mobility. This framework is called *MS-VAN3T* (**M**ulti-**S**tack framework for **VANET** applications testing in **ns-3**) and is presented in [47]. This project set itself apart from others by integrating, in one single open-source repository, all the models for access technologies offered in ns-3. In this way MS-VAN3T allows for easy switching between the said access technologies for an increased application development flexibility. The access models supported by MS-VAN3T are:

- 802.11p for V2V and V2I communications.
- LTE for V2N communications called LENA [48].

- C-V2X in transmission mode 4 for V2V communications [49].

On top of these access layer models, taking advantage of the flexibility that ETSI offers in this regard, an implementation of the ITS-G5 stack is provided by MS-VAN3T. This implementation is comprised by a model for the Application and Facilites layer together with BTP (Basic Transport Protocol) and GeoNetworking models (presented in this thesis). For the 802.11p model, BTP and GeoNetworking are implemented directly on top of the access layer while for the LTE and C-V2X models they work on top of UDP over IPv4 for the time being.

As described in the following sections the complete implementation of the ITS-G5 stack, specially for the 802.11p model, enables MS-VAN3T to expand the simulation applications towards the emulation domain. This is an innovative and exclusive feature present in MS-VAN3T that allows to send and receive ETSI standard-compliant packets from the real world through a physical interface.

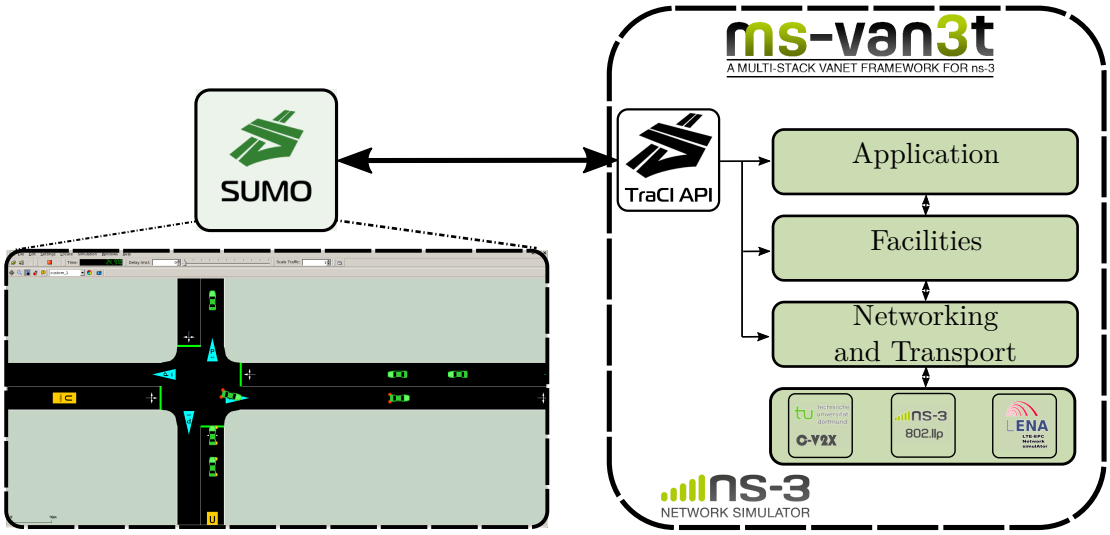


Figure 4.1: Main components of MS-VAN3T: on the left SUMO, allowing the mobility simulation and providing the GUI, which communicates via the the TraCI interface with MS-VAN3T in the ns-3 domain. MS-VAN3T implements the ITS-G5 stack over 3 possible access models: 802.11p, C-V2X in transmission mode 4 or LTE.

As presented in [47] MS-VAN3T works by coupling two simulators:

- SUMO (v1.8.0, at the time of writing): an open source traffic simulation suite which allows modelling of intermodal traffic systems, including road vehicles, public transport and pedestrians [50]. Furthermore it enables for interaction with the simulation elements making use of the TraCI API [51].

For a more user-friendly experience, SUMO offers a GUI (Graphical User Interface) enabling an interactive visualization of the simulation taking place.

- ns-3 (v3.33, at the time of writing): a discrete-event network simulator that takes care of all the communications between the entities involved in the simulation.

All messages exchanged in MS-VAN3T are ITS-G5 standard compliant and follow all the specifications described in Section 1.3.2 and further studied in Chapters 2 and 3. The framework architecture is shown in the Figure 4.1.

In this work we are not going to delve into the details of the Facilities layer implementation described in [47], that here can be considered as a black-box, enabling the transmission and reception of CAM and DENMs.

4.2 BTP model

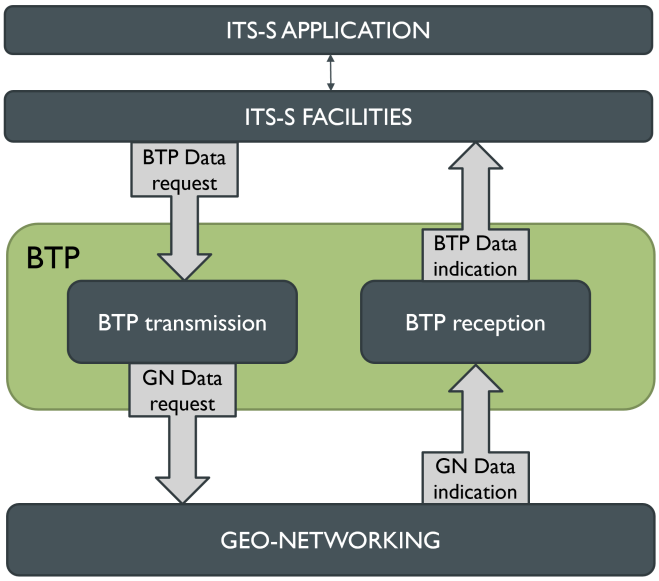


Figure 4.2: BTP layer implemented in MS-VAN3T.

As described in Chapter 2, the purpose of BTP is to enable the ITS Facilities layer to have access to the services of the GeoNetworking protocol by multiplexing and de-multiplexing the messages coming from both the CA basic service and DEN basic service (i.e., CAMs and DENMs). To achieve this purpose in the BTP model provided in MS-VAN3T the communication with the Facilities layers service is done through service primitives (described in detail on Appendix C). As shown in Figure

4.2, when a Facilities layer entity needs to send a message, it issues the so-called *BTP-Data.request* service primitive. In said *BTP-Data.request*, the ITS-FPDU (i.e., CAM or DENM) together with all the necessary control information for the creation of the BTP header is provided, such as the BTP type (BTP-B for both CAMs and DENMs), the destination port (2001 for CAMs or 2002 for DENMs) and the destination port information (set to zero by default). The control information is used by the BTP transmission module to create the BTP header that is then added to the ITS-FPDU to form the BTP packet (i.e., T-PDU). Once the T-PDU is created, the BTP transmission module issues the so-called *GN-Data.request* with the T-PDU and all the control information needed for GeoNetworking that was included in the *BTP-Data.request*.

On reception, the BTP reception module extracts the BTP header from the T-PDU found in the *GN-Data.indication* according to BTP type specified in the latter (BTP-B). With the destination port field of the BTP header, the BTP reception module selects the correct Facilities entity and issues a *BTP-Data.indication* with the ITS-FPDU of the T-PDU together with the control information needed by said entities that was included in the *GN-Data.indication*. The code of both modules of the BTP model implementation is reported in Appendix A.

4.3 GeoNetworking model

Going down in the ITS-G5 stack of MS-VAN3T, we find the GeoNetworking (GN) model which has two main modules, as shown in Figure 4.3: the *network management* module and the *packet handling* module.

4.3.1 Network management module

The network management module is the one responsible of managing all the data structures of the protocol that need to be handled regardless of the packet handling procedures dictated by the upper protocol entities. In order to do that, the network management module relies on a Vehicle Data Provider (VDP) to provide all the vehicle's information such as position, speed, and heading. In MS-VAN3T the VDP is linked to the TraCI interface which provides the information from the simulated vehicles in SUMO. This does not mean that MS-VAN3T can only make use of TraCI; indeed, the modularity of MS-VAN3T allows the easy integration with other type of mobility traces, for example coming from real GPS devices. The network management module features of MS-VAN3T include:

- Address configuration
- Ego position vector update

- Beaconing

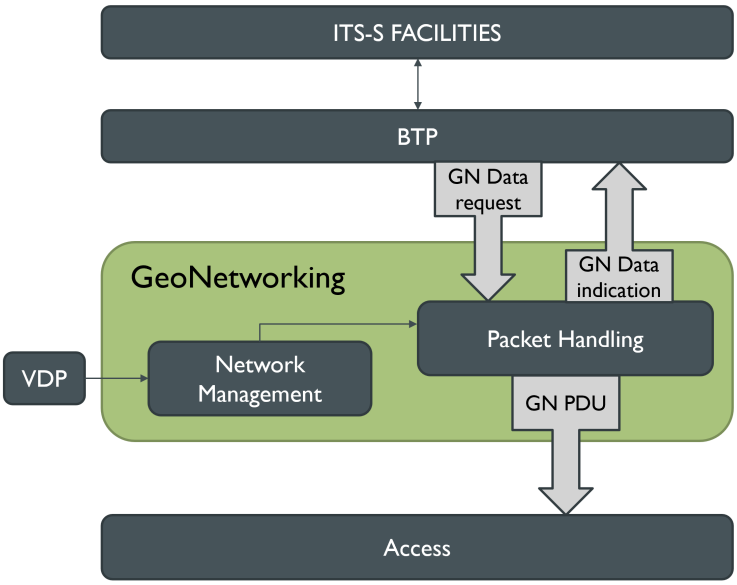


Figure 4.3: GeoNetworking layer implemented in MS-VAN3T.

Of the address configuration methods, described in Section 3.2, the managed configuration method is adopted as default in the GN model. At start-up the network management module creates a new GN address with the corresponding ITS-Station (ITS-S) type and MID fields, the latter set with the 48-bit MAC layer address.

The ego position vector, described in Section 3.3.2, is updated by the network management module every second with the information provided by the VDP. In case of a stationary ITS-S (e.g., a *roadside unit*) the EPV is only set at start-up by the application with the ITS-S's position (i.e., longitude and latitude) and with the speed and heading set to zero.

In order to periodically advertise the GeoAdhoc router's position to its neighbors, the beaconing mechanism is used. The beaconing mechanism consists in sending a periodic BEACON packet (described in Section 3.4.5) unless another GN packet that carries the GeoAdhoc router's EPV (e.g., GBC and SHB) is sent, which results in a re-schedule of the BEACON packet. When a new BEACON packet needs to be sent, the network management module creates a new *GN-Data.request* for the packet handling module to handle as explained next.

4.3.2 Packet handling module

The packet handling module takes care of the packet transmission and reception. The main packet handling features present in MS-VAN3T are:

- BEACON packet handling.
- SHB packet handling.
- GBC packet handling.

Packet transmission

When a *GN-Data.request* is received from the BTP entity, the packet handling module creates a new GN packet (i.e., GN-PDU). According to the information found in the data request, both basic and common headers, are created in the same way for all GN packet types. Starting from the basic header (described in Section 3.4.2) settings, the protocol version is always set to 1 as well as the next header (i.e., common header), given that no security features are yet present in MS-VAN3T. The lifetime and remaining hop limit fields are set with the *Maximum Packet Lifetime* (MPL) and *Maximum Hop Limit* (MHL) parameters found in the *GN-Data.request*.

Following with the common header, its next header field is set with *Upper protocol* parameter in the *GN-Data.request* which for both CAM and DENM is set as BTP-B and for BEACON packets is set as unspecified. Afterwards, the header type is set with the *Packet transport type* parameter in the *GN-Data.request* while the header sub-type is set to 0 for BEACON and SHB packets or to the value of the specified geographical area shape. The traffic class, maximum hop limit and payload are set with their homonym parameters of the *GN-Data.request* while the mobile flag is set to 1 if the ITS-S is mobile or to 0 otherwise.

For each GN packet type the GN entity selects the transmission procedure which will create the corresponding extended header to be added to the T-PDU, followed by the common and basic headers. All GN packet types available in MS-VAN3T have the Long Position Vector (LPV) in their extended headers, which is set with the values found in the EPV and the GN address before proceeding with the selected transmission procedure. For BEACON packets the extended header is created consisting only in the LPV similarly to SHB packets, with the difference of a media-dependent field set to zero by default in the latter. Lastly, in addition of the LPV, the GBC extended header is also comprised with the sequence number incremented on each GBC packet sent, and the geographical area to which the packet (DENM) is addressed to, found in the *GN-Data.request*. Once all headers are set and added to the T-PDU (with the exception of the BEACON, which has no T-PDU in it) to create the GN-PDU, the latter is sent to the lower layer which

depends on the access technology model being used. In the case of 802.11p the GN-PDU is sent to the Logical Link Control (LLC) sub-layer while in the case of either LTE or C-V2X models the GN-PDU is further encapsulated in UDP and IPv4, and sent to the underlying LTE stack.

Packet reception

Now changing the perspective to the receiver side, on reception of GN-PDU from the lower layers the packet handling module extracts all GN headers and parse the information to be sent together with the T-PDU encapsulated in the *GN-Data.indication*. Starting from the basic header, the protocol version is checked to proceed if the received packet belongs to the same version of the protocol implemented in MS-VAN3T or to drop the packet otherwise. If specified so in the basic header in its next header field, the common header processing starts. After checking that the maximum hop limit is indeed higher than the remaining hop limit specified in the basic header, all fields of the common header are set in their respective fields of the *GN-Data.indication*. According to the GN type and sub-type fields, the BEACON, SHB or GBC processing is selected.

For BEACON and SHB packets a very similar procedure is followed, after extracting the LPV from the extended header, the *source LPV* parameter of the *GN-Data.indication* is set. Then, if the local GeoAdhoc Router's local GN address was set with the auto-address configuration, with the GN address of the LPV, the Duplicate Address Detection (DAD) takes place. The DAD, as introduced in Section 3.2, is used to check if the GN address found in the received packet's LPV coincides with the one of the local GeoAdhoc router. If a duplicate address is detected, a new MID field is set for the local GN address. Next, in the local Location Table (LocT, described in Section 3.3.1), if an entry (LocTE) exists for the source's GN address, it gets updated or a new entry is created with the values of the LPV otherwise. Given the fact that both SHB and BEACON packets are always received by neighbor nodes only, the *IS_NEIGHBOUR* flag of the LocTE is set to TRUE. Lastly, in case of a SHB packet, the *data* parameter of the *GN-Data.indication* is set with the T-SDU obtained after the extraction of all headers and passed on to the BTP layer.

For GBC packets, after setting the source LPV parameter of the *GN-Data. indication* with the LPV in the GBC header, if a LocTE exist for the source LPV, the Duplicate Packet Detection (DPD) takes place. The DPD is carried out with the use of the sequence number extracted from the GBC header in order to detect possible duplicates created because of forwarding procedures not yet supported in MS-VAN3T, but ready to be implemented in future work. After the DPD, the DAD and the update of the LocTE (if it exist) are executed as explained for BEACON and SHB packets. Lastly, before passing the *GN-Data.indication* to the BTP layer,

the GeoAdhoc Router checks if it is inside the geographical area extracted from the header. If it is outside the area, the packet is dropped.

In order to check if the GeoAdhoc router is inside the geographical area which shape is specified in the common header GN sub-type field. In [46] a set of functions $F(x, y)$ are defined to be used by an ITS station to determine whether a point $P(x, y)$ is located inside the area. The functions $F(x, y)$ have the following possible results:

- $F(x, y) = 1$ for $x = 0$ and $y = 0$: vehicle at the center point of the geographical area.
- $F(x, y) > 0$: vehicle inside the geographical area.
- $F(x, y) = 0$: vehicle at the border of the geographical area.
- $F(x, y) < 0$: vehicle outside the geographical area.

The functions $F(x, y)$ represent the canonical form of the geometric shapes, i.e., the Cartesian coordinates has their origin in the center of the shape. The Cartesians coordinates of the point $P(x, y)$ represent the vehicle's position in relation with the center of the geographical area.

To compute $P(x, y)$, the longitude and latitude of the GBC header fields are first converted to Cartesian coordinates and substracted from the longitude and latitude of the EPV (Ego Position Vector), previously converted to Cartesian coordinates as well. In this way if the coordinates of the EPV are the same to that of the geographical area's center, then x and y will be equal to zero. Afterwards, depending on the shape of the geographical area indicated in the *Header Sub-Type* field of the Common Header, a different function $F(x, y)$ is computed.

For a circular area the following function $F(x, y)$ is used:

$$F(x, y) = 1 - \left(\frac{x}{r}\right)^2 - \left(\frac{y}{r}\right)^2$$

where x and y are the coordinates of $P(x, y)$, and r is the value of the `"distance a"` field of the GBC header.

For a rectangular or ellipsoidal area, before computing $F(x, y)$, some considerations have to be made for the possible rotation of the shape from the north axis. To compensate this rotation, a clockwise rotation around the origin (i.e., the center of the shape) is applied to $P(x, y)$. The idea is to obtain the projected $P(x, y)$ in a rectangular or ellipsoidal area with their long side perpendicular to the north (i.e., to the ordinate axis) as shown in Figure 4.4.

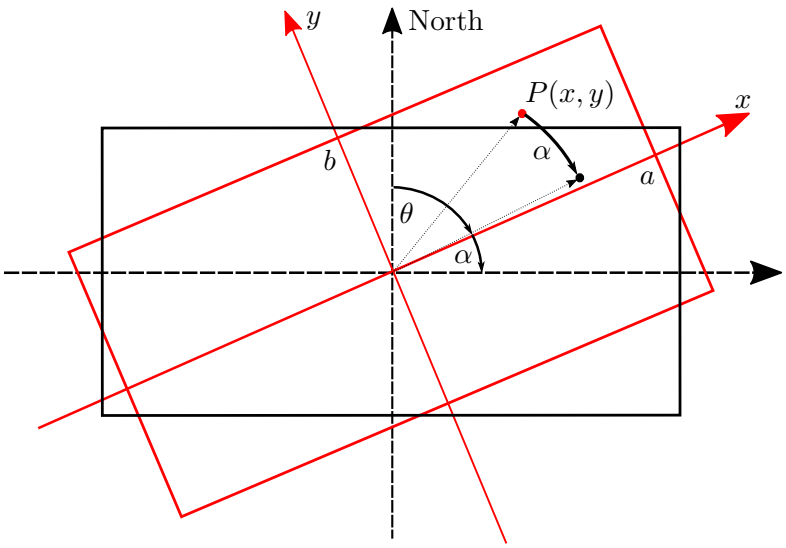


Figure 4.4: Geographical area rotation compensation.

In order to obtain this new $P(x, y)$, the angle used to rotate is the so-called *zenith* angle (i.e., α angle shown in Figure 4.4 given by $\alpha = 90 - \theta$.

After compensating the rotation, the following $F(x, y)$ are used:

- For rectangular area:

$$F(x, y) = Minimum\left(1 - \left(\frac{x}{a}\right)^2, 1 - \left(\frac{y}{b}\right)^2\right)$$

- For ellipsoidal area:

$$F(x, y) = 1 - \left(\frac{x}{a}\right)^2 - \left(\frac{y}{b}\right)^2$$

where x and y are the coordinates of $P(x, y)$, while a and b are the values of the `"distance a"` and `"distance b"` fields of the GBC header respectively.

The code of both modules of the GN model implementation together with the complete list of features is reported in Appendix B.

4.4 MS-VAN3T communication scenarios

4.4.1 V2I/V2N scenarios

As described in [47], MS-VAN3T provides V2I and V2N application models, in which vehicles are configured to periodically broadcast CAMs (i.e., SHB packets) that are collected by a server who can send DENMs (i.e., GBC packets) back to

the vehicles if the application logic commands it. The centralized scenarios can be enabled using two different communication technology models that depending on the selected one, the resulting scenario will be considered either V2I or V2N. These two communication technologies models are:

- 802.11p, using the WAVE model available in ns-3. In this case, vehicles are equipped with OBUs (On Board Units), and communicate with a remote host that is placed behind a RSU (Road Side Unit). Due to the proximity between the clients and the server, and due to the fact that the clients and server are in the same subnet, we can define it as a V2I communication. The scenario using this access model is shown in Figure 4.5.

- LTE, using LENA [48] as simulation framework, in which a classic LTE network is established, with the vehicles acting as UEs (User Equipment) connected to the eNB (eNodeB), that is in turn connected to the EPC (Evolved Packet Core) through the S1-U interface. The EPC implements the SGW (Serving Gateway) and PGW (Packet Data Network Gateway) blocks. The PGW is connected to a remote host that runs the application server logic. This communication can be seen as V2N, since the UE is connected to a service provider that can be ideally everywhere in the Internet. The scenario using this access model is shown in Figure 4.6.

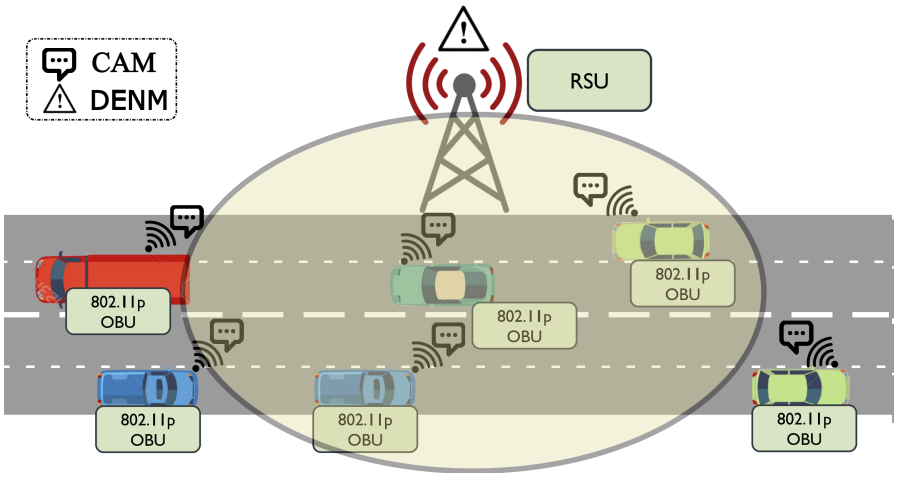


Figure 4.5: Centralized scenario with 802.11p as access model. CAMs are sent through vehicle's 802.11p OBU towards the to the RSU, where the server is directly connected.

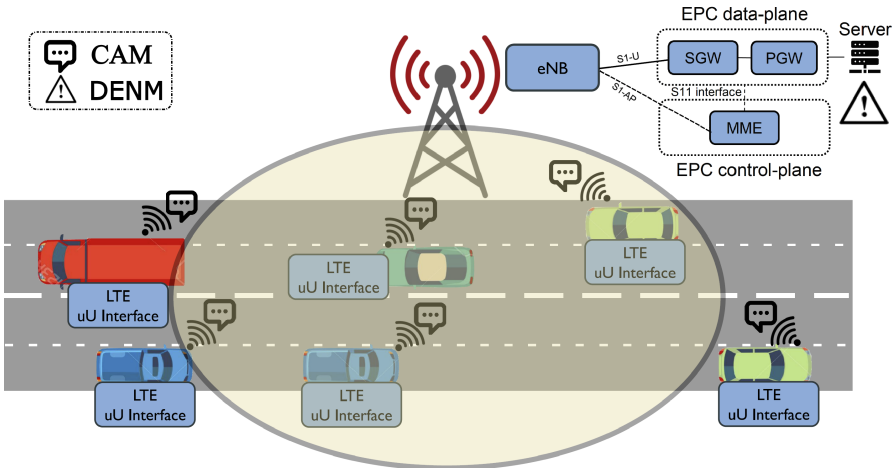


Figure 4.6: Centralized scenario with LTE as access model. CAMs are sent through the Uu interface towards the server, which is connected to the EPC via the PGW.

4.4.2 V2V scenarios

To complete the available set of scenarios provided by MS-VAN3T, the V2V models are presented as described in [47]. In this particular configuration, the vehicles are configured to broadcast CAMs, that are received by all vehicles in communication range (often referred to as neighbors). On a CAM reception, the vehicles run their own application which will dictate the trigger of DENMs communicated in a completely distributed fashion. These scenarios are enabled by the following communication models:

- C-V2X, using the framework proposed by Eckerman et al., [49], in which a C-V2X Mode 4 network is modeled. In this model, the vehicles communicate through the PC5 interface and directly exchange messages with their peers through sidelink (SL), without relying on the eNB.

- 802.11p, using the WAVE stack available in ns-3. As in the V2I scenario, vehicles are equipped with OBUs, but in this case they broadcast CAMs and DENMs among themselves.

4.5 MS-VAN3T sample applications

With the intention of showcasing the potentially of the framework, MS-VAN3T comes with two main sample applications presented in [47]. The first one called

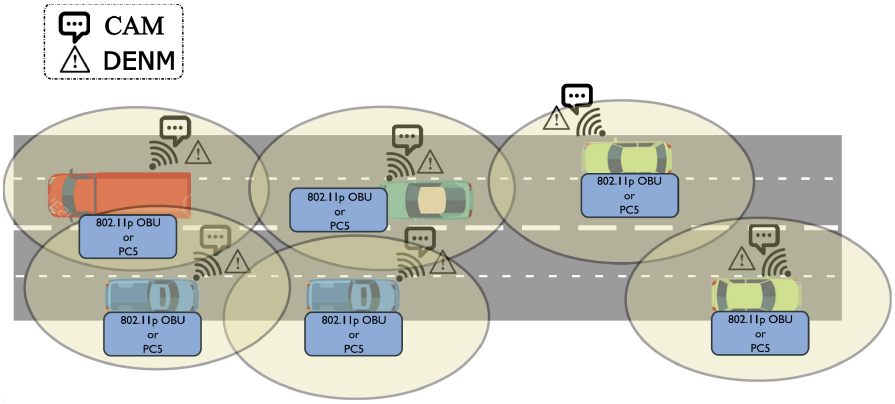


Figure 4.7: Distributed scenario for both access models: 802.11p and C-V2X. CAMs and DENMs are sent by vehicles either through their OBUs or PC5 interface.

area speed advisory, showcases the V2I/V2N scenario working under a centralized client/server architecture while the second one, called *emergency vehicle alert*, showcases the V2V scenario working under a completely distributed manner.

In addition to these main applications, MS-VAN3T offers an emulation application on which, relying on a physical interface, is possible to send the CAMs and DENMs generated by the simulated vehicles over a real network.

4.5.1 Area speed advisory

The map used in this context, shown in Figure 4.8, includes two road crossings connected through a central two-way street. The network access point (eNB for LTE and RSU for 802.11p) is placed at the center of the map, configured to have sufficient coverage to provide connectivity to all vehicles traveling in the simulated area.

The idea of the application is to divide the map into two areas, each one with different maximum speeds allowed. The first area is located in the center of the map surrounding the network access point (that from the application's point of view, we can call server) and its maximum allowed speed is set to 25 km/h. The second area comprising the rest of the simulated map has a maximum allowed speed of up to 100 km/h. The CAM and DENM exchange works differently depending on the access technology model selected due to the LTE model constraint for broadcasting capabilities.

For the 802.11p model vehicles are configured to periodically broadcast CAMs. As soon as the server (i.e., RSU) receives a CAM, it immediately triggers a DENM to be geographically broadcasted to all vehicles (i.e., OBUs) within the low speed area, to warn drivers about the necessity of reducing their speed. These DENMs

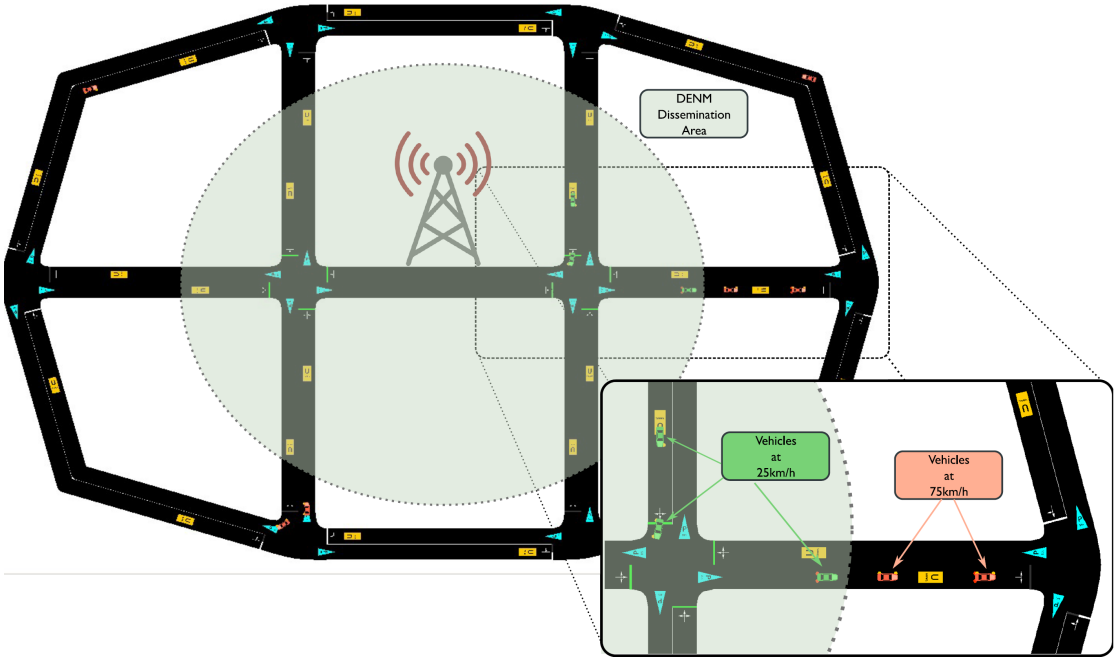


Figure 4.8: Two screenshots from SUMO-GUI, showing the map and the V2I/V2N application in action. The center area is denotes the slow speed area.

are set to be periodically updated each second unless the server has not received a CAM in the last five seconds, indicating that no vehicles are traveling in the low area, to decrease the channel load. In the case that a vehicle in the low speed area has not received a DENM message for one and half seconds, it understand that actually it has got out of the area and is now allowed to increase its speed. This mechanism is in big part enabled by the GN entity which drops GBC packets (i.e., DENMs) when the vehicle is outside the low speed area specified in the GBC header, avoiding for the application to receive them.

On the other hand for the LTE model, in which each message is transmitted using UDP/IPv4-based stack, only unicast messages are allowed and that translates in some differences. In this case the server monitors the position of the vehicles, by reading the CAMs, and warns them if they enter an area with different speed restrictions. Vehicles are configured to periodically send CAM messages, which are received by the network access point and forwarded to the server. The server, aware of the boundaries of the two areas, analyzes the position of the vehicles. Whenever it realizes that a vehicle is moving from a zone to another, it sends a DENM message, warning the driver about the necessity of reducing his/her speed or the about possibility to increase it. In this scenario, CAMs are sent as unicast from the vehicles to the server running the application logic. At the same time, the

server generates and sends unicast DENMs to notify the vehicles changing speed area [47].

4.5.2 Emergency vehicle alert

In this application the map used, as can be seen in Figure 4.9, consist of a circular road with two lanes for each direction. Vehicles are configured to have a maximum speed varying between 30 km/h and 60 km/h. This application, in addition to normal vehicles (referred to as *passenger car* by ETSI [42]), introduces a new kind of vehicle called *Emergency Vehicle* (EV). These EVs are intended to act as an ambulance, a police car or even a firefighter truck. In *emergency vehicle alert* EVs are configured to have a maximum speed up to 75 km/h allowed to simulate an emergency behavior.

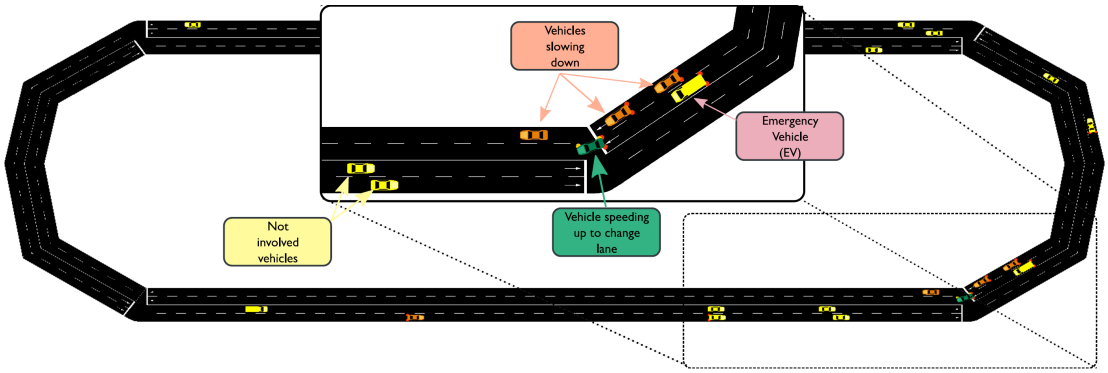


Figure 4.9: Two screenshots from SUMO-GUI, showing the map and the V2V application in action.

Moving from the mobility aspect and focusing on the communication details , in this application every vehicle is set to periodically broadcast CAMs to all its neighbors (i.e., all vehicles in communication range). When passenger cars receive a CAM from an emergency vehicle (CAMs with the *StationType* Data Element set to *specialVehicles*), they will check the distance between them and the heading angle difference to the one of the emergency vehicle. In case both the heading angle difference and the distance between them falls under a preconfigured threshold, the passenger car understands that the EV is approaching. Under this circumstances, if the passenger car is traveling on the same lane as the EV, it will slow down. On the other hand, if it is on a different lane, it will change lanes as soon as it can (even speeding up if needed).

Since both access technology models (802.11p and C-V2X) allow broadcast communication, the application logic is the same in both cases.

4.5.3 Emulator application

Last but not least, the emulator application available on MS-VAN3T showcases the ability of the framework to send ETSI ITS-G5 standard-compliant messages out to the real world over a real network as shown in Figure 4.10. At the time of writing, this sample application is relying on the same map and mobility traces of the V2V application and it sends both CAM messages and periodic DENM messages, shown in Figure 4.11 and 4.12, as an example on how both kinds of messages can be emulated and sent to the external world.

More in details, this application emulates N vehicles, each with its own CA and DEN basic service, and make them send the CAM/DENM messages and receive the CAM/DENM messages through a physical interface. These capabilities of MS-VAN3T would enable, in the future, hardware-in-the-loop testing and evaluation since the framework is able not only to send but to receive and dissect ETSI standard compliant packets.

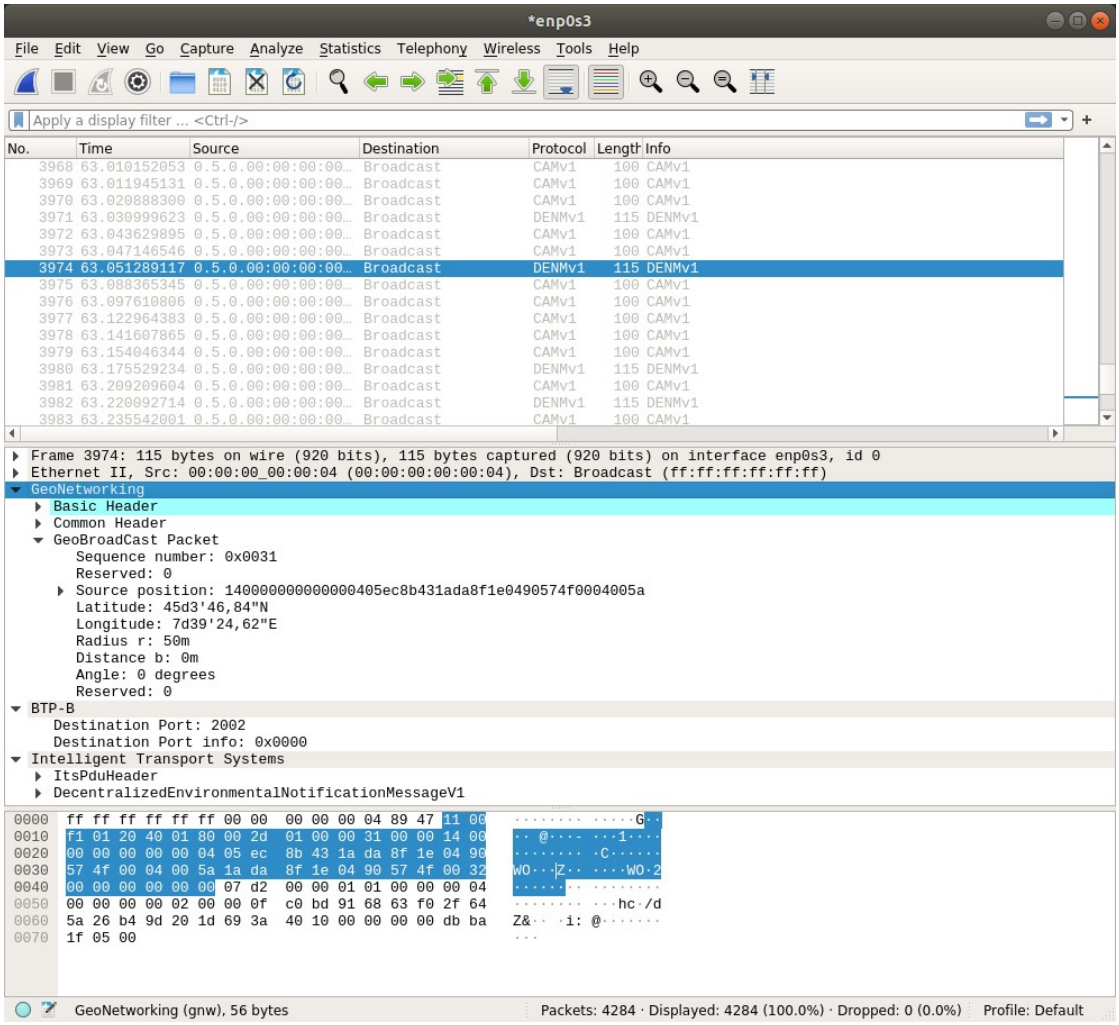


Figure 4.10: Screenshot of a Wireshark capture of the messages sent by the emulator application, selecting the enp0s3 interface.

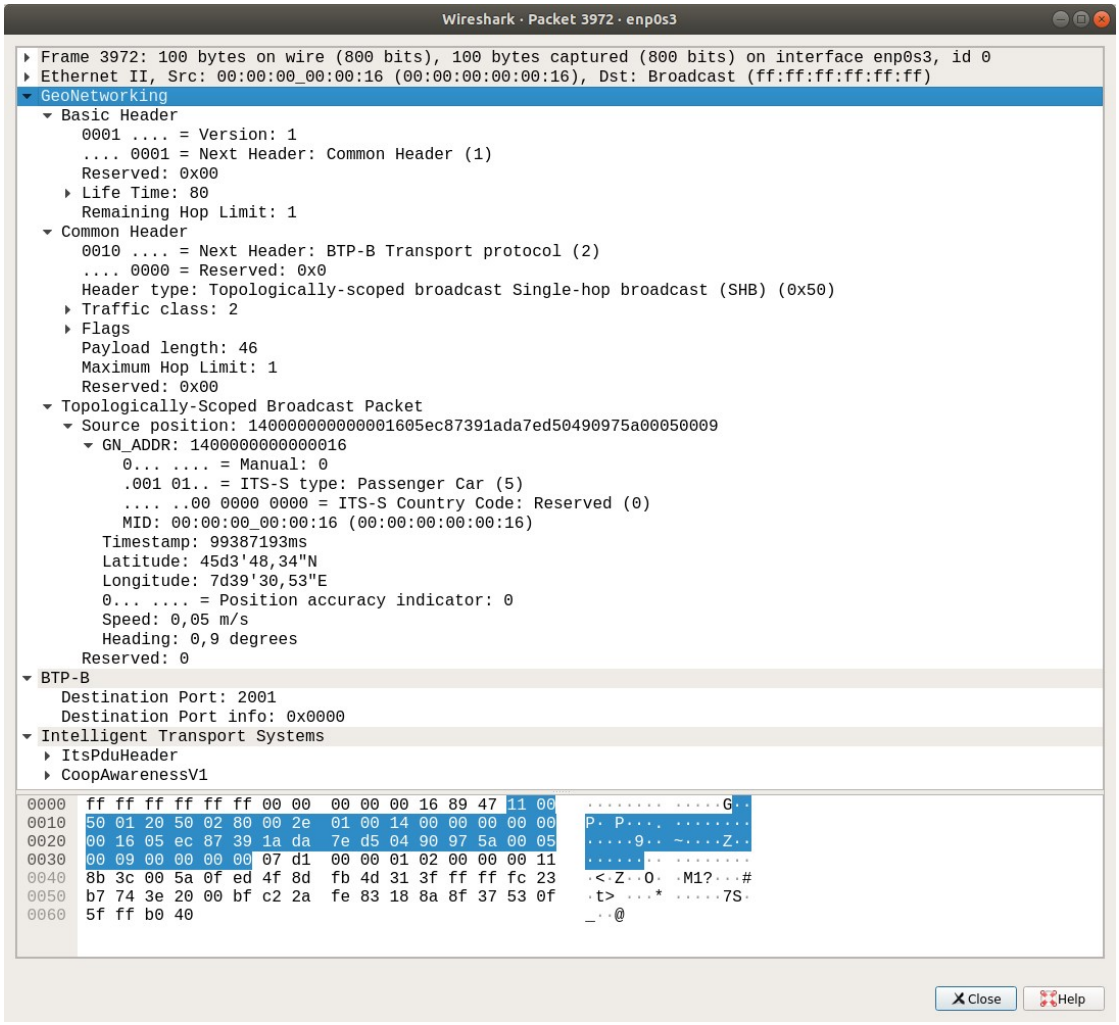


Figure 4.11: Screenshot of a Wireshark capture of a CAM (i.e., SHB) sent by the emulator application.

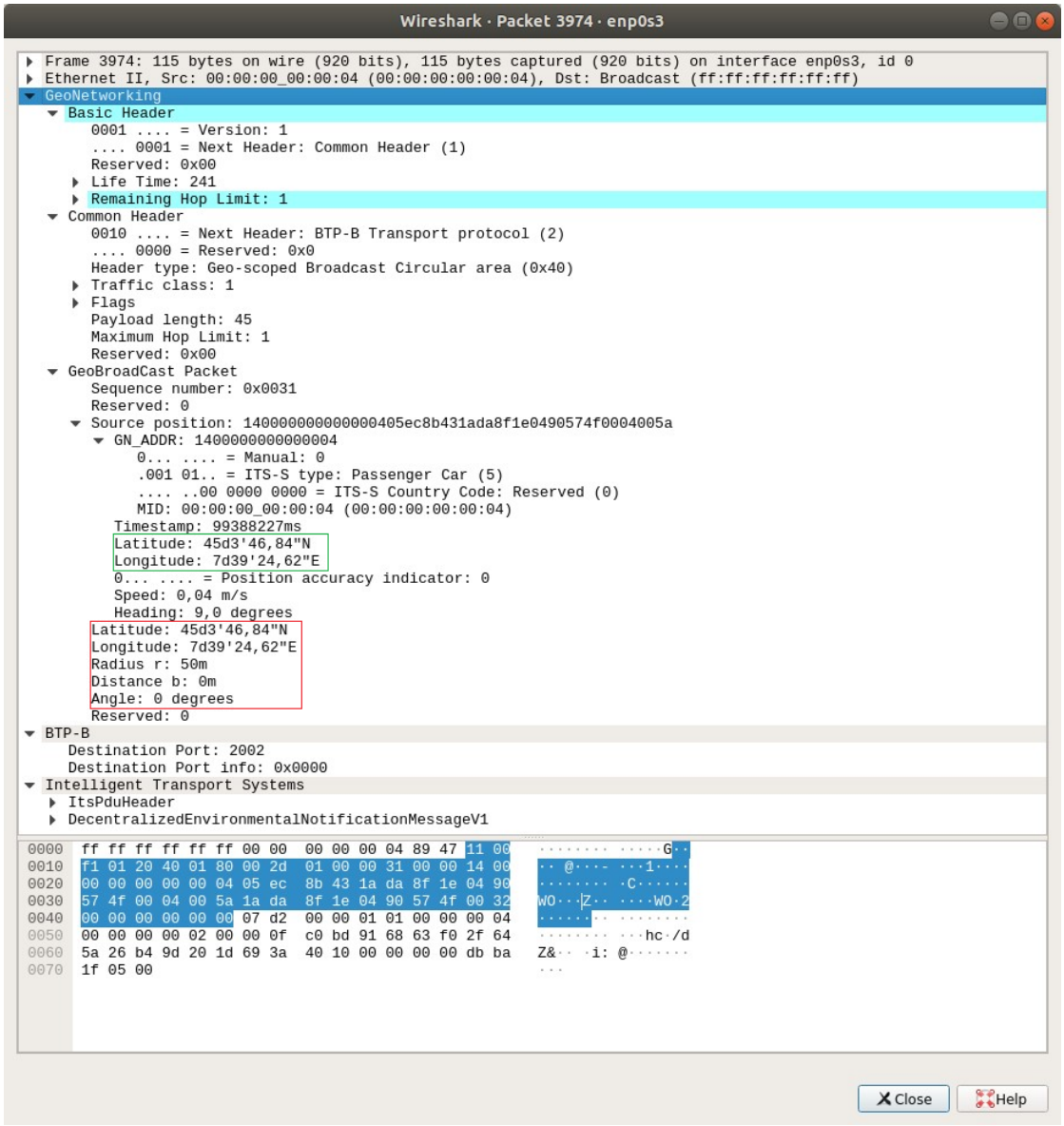


Figure 4.12: Screenshot of a Wireshark capture of a DENM (i.e., GBC) sent by the emulator application. Here the fields defining the addressed geographical area are highlighted in red, which defines a circular area with a radius of 50 [m] and centered on the vehicle's position (highlighted in green).

Chapter 5

Application testing on MS-VAN3T

As described in the previous chapter, MS-VAN3T offers a flexible tool that can be used to easily develop and test a vast number of applications. As part of this work, a new application has been developed and tested under the MS-VAN3T framework. This chapter describes this new V2X service and its evaluation.

5.1 Traffic Manager

In addition to traffic safety, traffic efficiency applications are of great interest for ITS and it is not a novelty that congestion due to traffic lights can constitute a major problem, specially for big cities. Traffic congestion not only causes traffic efficiency issues resulting in longer travel times but, as described in Chapter 1, also environmental problems due to increased vehicle emissions. The main limitation of the conventional traffic lights used to coordinate traffic on road intersections comes from the fact that their switching periods are fixed, not taking into account eventual changes in the traffic flow. Traffic flow is not static and can vary depending on a great number of factors that are very difficult to predict and therefore make static patterns implemented in traffic lights very inefficient. The idea of the *Traffic Manager* application is to make traffic lights, commanded by a RSU, able to withstand changes in the traffic flow present on an intersection.

For this application, the map used is the same one present in the *Area Speed Advisory* sample application, described in Section 4.5.1. Similarly to *Area Speed Advisory*, the *Traffic Manager* application works by relying on V2I communications. The main difference is that the traffic lights, placed in the two main intersections, are configured to have two phases as shown in Figure 5.1 allowing North-South and East-West traffic flows respectively. Additionally, for each phase, a yellow light

transition phase is set to warn vehicles before a red light.

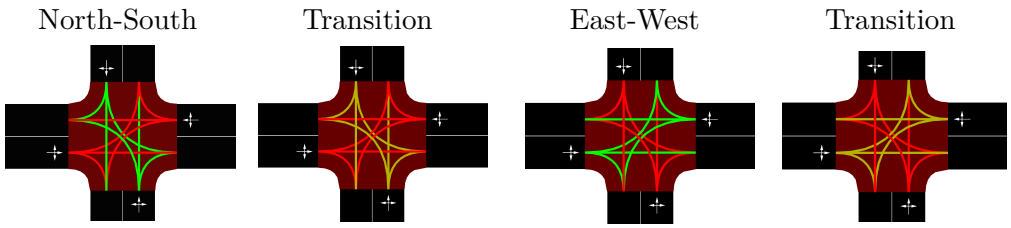


Figure 5.1: Lights phases in the Traffic Manager application.

Vehicles are set to periodically broadcast CAMs that are received by the RSU which in turn, using the geographical position and heading angle extracted from the CAM, determines to which traffic light the vehicle is going to respond to. This information is used to update the *intersection state* that contains the number of vehicles belonging to each traffic flow. At start-up the duration of both phases are set as half of the value of the complete cycle (without the transition phases) that is set to 40 seconds by default and can be otherwise set by the user. After the first cycle, the duration of the phases for the following one are set depending on the *intersection state* as follows :

$$phase_1 = cycle * \left(\frac{NS}{NS + EW}\right)$$

$$phase_2 = cycle * \left(\frac{EW}{NS + EW}\right)$$

where NS and EW are the North-South state and East-West state respectively and $NS + EW$ is the total number of vehicles at the intersection.

Given the case that the number of vehicles of a traffic flow exceeds a threshold value, set by default to 70% of the total number of vehicles at the intersection (enabled to be set by the user to a different value), and the current phase does not belong to that traffic flow, the cycle is preempted as seen in Figure 5.2. When the cycle is preempted the duration of the phases are recomputed, as described, with the updated values of the *intersection state*.

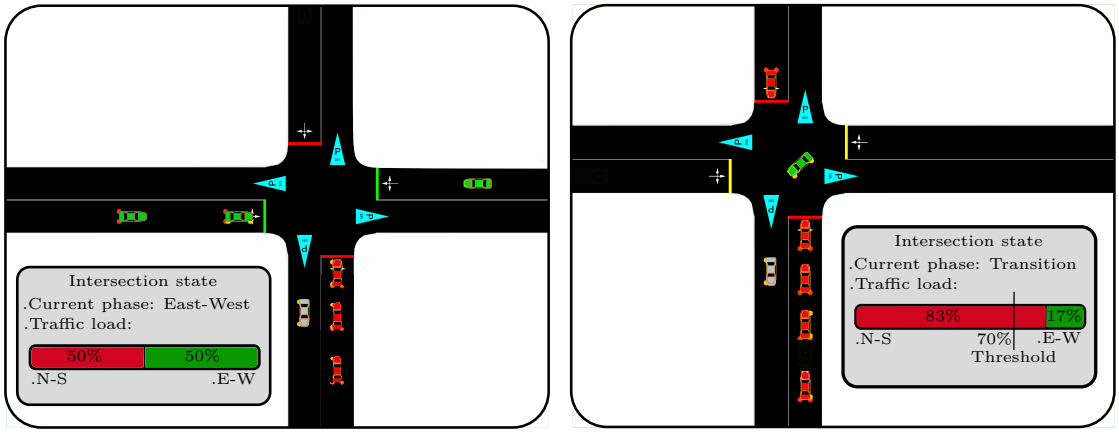


Figure 5.2: Two different intersection states. On the left, a balanced intersection state with equal traffic flow loads. On the right, an unbalanced state where the North-South flow load has surpassed the threshold of 70% and the cycle has been preempted and currently on transition phase.

5.1.1 Application performance

In order to test the *Traffic Manager* application performance, simulations have been carried out to measure the average speed of all vehicles as an indicator of traffic efficiency, the main objective of the application. The first simulations were configured to run without the application where traffic lights phases were static and, for comparison, different cycle durations have been tested. The tests involved an increasing number of vehicles, ranging from 15 to 70 vehicles, which were configured to have an allowed maximum speed of up to 75 km/h. All the tests consisted on 10 simulation runs of 1 hour of simulated time each. The results are plotted in Figure 5.3, where the increasing number of vehicles is reported on the x axis and the measured average speed of all vehicles on the y axis. As can be seen, when the number of vehicles is small, at a lower cycle duration the average speed is higher given the nonexistence of traffic congestion. When the number of vehicles is increased and so does the traffic congestion, the results become very similar for all cycle duration with a pronounced decrease in the average speed of vehicles.

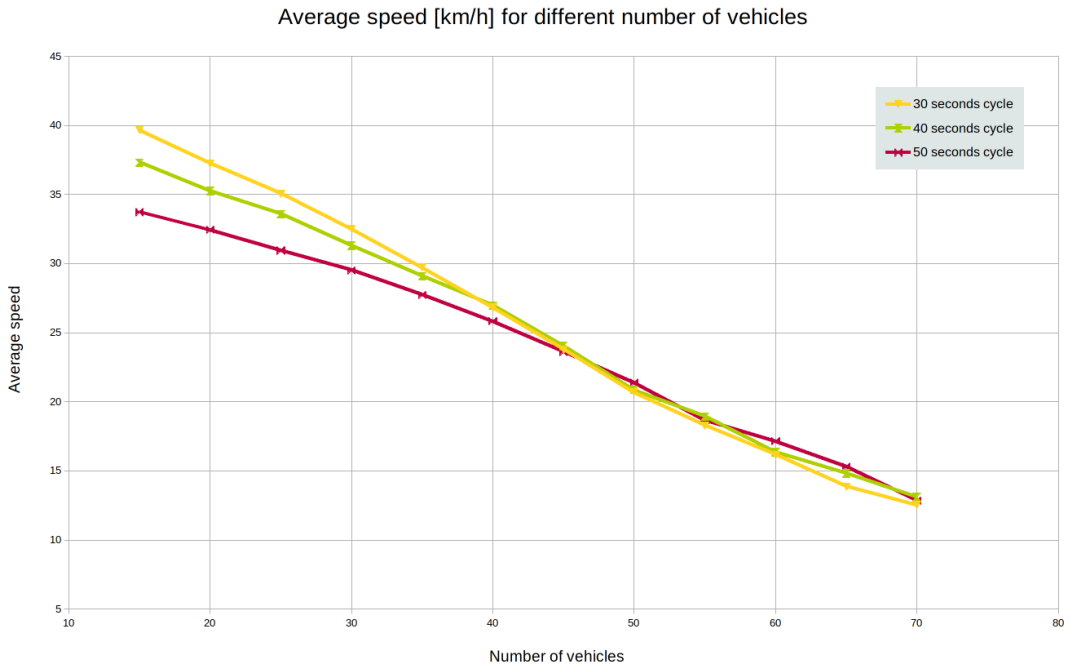


Figure 5.3: Results for simulations with Traffic manager disabled for different cycle duration settings. The plot shows the average speed of all vehicles as a function of the number of vehicles present in the simulation.

On the other hand the simulations carried out to test the application in action, plotted in Figure 5.4 with the increasing number of vehicles reported on the x axis and the measured average speed of all vehicles on the y axis, showed rather different results. For the application tests, simulation runs were configured with different cycle durations and different threshold values to compare different performance. As depicted in Figure 5.4, all the tests under different configurations show almost identical performances both at high traffic congestion, as it happened without the application, and at low traffic congestion, product of the preemption mechanism that is more active when the number of vehicles at intersections is smaller. In the same fashion as the first tests, all results are obtained from 10 simulations runs of 1 hour of simulated time each.

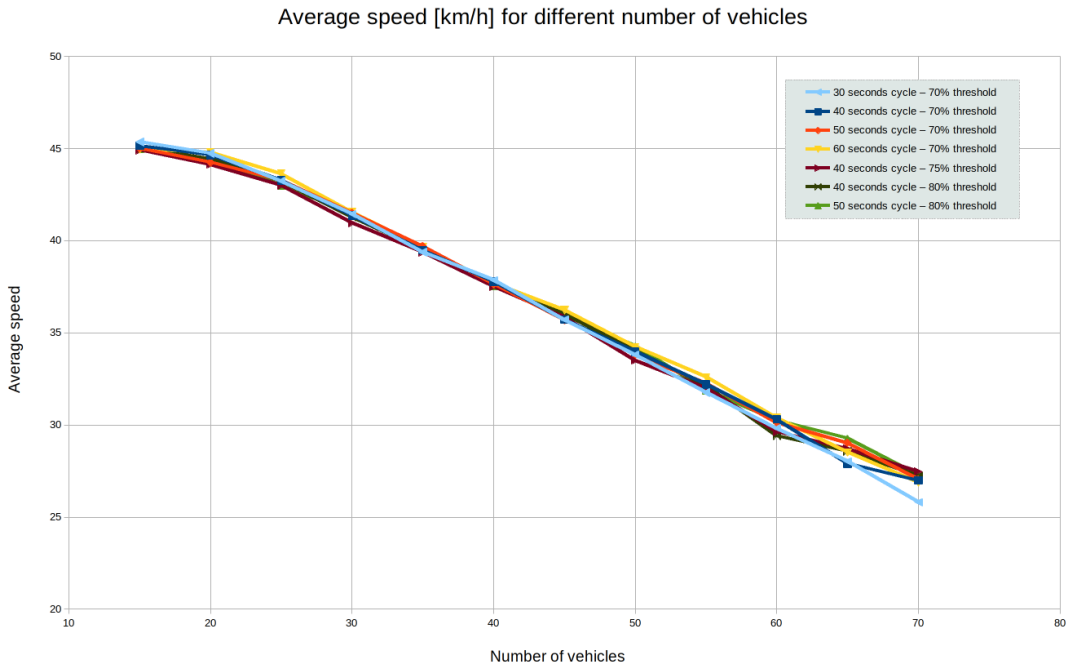


Figure 5.4: Traffic manager application results for different cycle duration and different threshold values. The plot shows the average speed of all vehicles as a function of the number of vehicles present in the simulation.

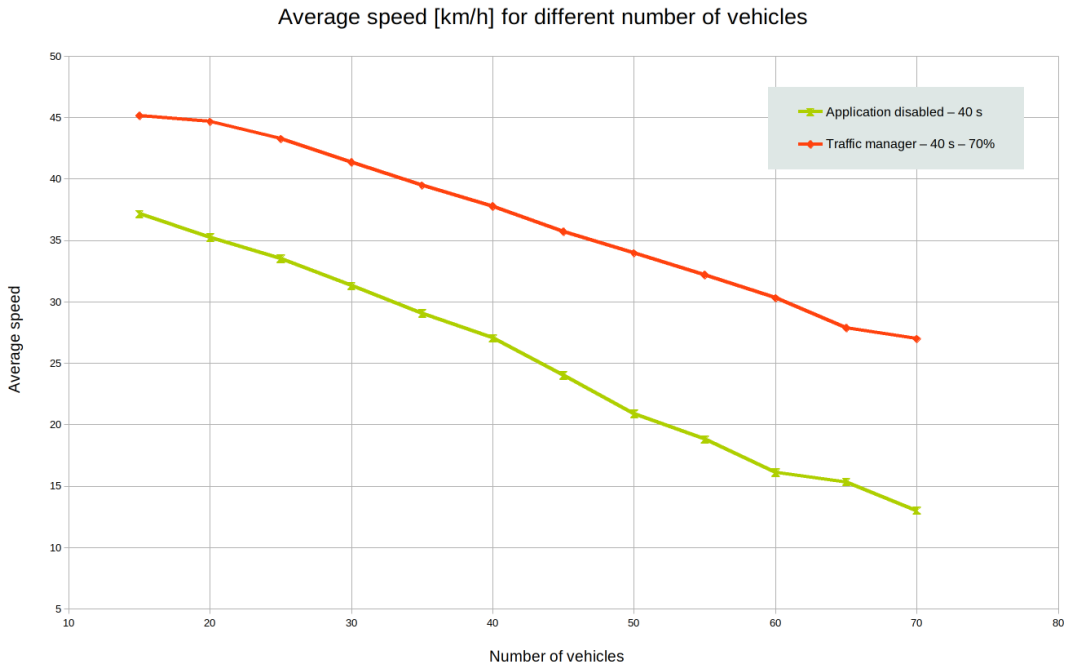


Figure 5.5: Traffic manager application results compared against results with the application disabled. For Traffic manager the cycle was set to 40 seconds and the threshold to 70% and for the results with no application, the cycle was set to 40 seconds as well. The plot shows the average speed of all vehicles as a function of the number of vehicles present in the simulation.

Lastly, a comparison of the results of the tests with and without the *Traffic Manager* application is reported in Figure 5.5, showing the performance with a 40 seconds cycle duration for both and with a threshold of 70% for the one with the application enabled. The results show the introduced benefits of the *Traffic Manager* application in the measured average speed of vehicles, resulting in a improvement of traffic efficiency from 20% on low traffic congestion and up to 90% on high traffic congestion. This new application developed on top of MS-VAN3T, shows the potentiality of the framework to enable new vehicular applications of all kinds to be easily and efficiently tested under different scenarios.

Chapter 6

Conclusions

A world characterized by Intelligent Transport Systems will be soon a reality and each day new steps in the development of fully connected vehicles, that are the key feature of ITS, are made by automotive industries, government entities and researchers alike, all over the globe. To make this possible, during the last two decades new technologies for vehicular communications have come along followed by standardization efforts to enable the coexistence of them so that new ITS applications may be developed taking advantage of all their features. At the beginning of this thesis, a description of the current landscape of vehicular networks is made, going through a brief analysis of the different access technologies and protocols in this context and taking a deeper look into the ETSI ITS-G5 set of protocols.

With the aim of deepening the analysis of ITS-G5, on Chapters 2 and 3 an extensive description of the Networking and Transport layers is made to further analyze the details of the BTP and GeoNetworking protocols. The idea behind the aforementioned chapters is not only to understand the set of features these protocols provide to cope with VANETs needs, but also to describe in detail how these protocols interact with higher and lower layers of the ITS-G5 stack, and the packet structure used for their operation.

Going into the main objective of this thesis, the MS-VAN3T framework is introduced, upon which all the work has been carried out. As described, MS-VAN3T provides an implementation of the ETSI ITS-G5 stack, initially comprised by a model for the Application and Facilities layer, and which has been further extended in this work to include a model of the BTP and GeoNetworking protocols to complete the stack with the available access technologies models already present in the framework.

These new introduced models enable MS-VAN3T to expand its already powerful functionalities for the development and testing of new ITS applications, allowing to fully exploit the capabilities of the CA and DEN services. This enhanced

capabilities are showcased in the sample applications provided by the framework, that benefit from the GeoBroadcast and Single-Hop Broadcast features provided by the GeoNetworking model. Furthermore, the ability to send ETSI ITS-G5 standard-compliant messages allows MS-VAN3T to be not only a simulation tool but also an emulator able to send messages from the simulated vehicles into the real world with a physical interface and to receive and dissect standard compliant packets for future hardware-in-the-loop testing.

MS-VAN3T has been envisioned to allow researchers to easily and efficiently develop new vehicular applications, and with the intention to showcase this potential, a *Traffic Manager* application has been developed. As detailed in Chapter 5 this application proves how the traffic efficiency issues caused by conventional traffic lights can be significantly reduced by allowing them to be controlled by a roadside unit able to receive CAMs from nearby vehicles.

6.1 Future work

The GeoNetworking model implemented for this thesis does not feature all the functionalities specified by ETSI and is only intended to provide the ones needed for the Facilites layer models implemented in the framework to fully exploit their services. As mentioned throughout the thesis some functionalities from the protocol have been left out for future work such as forwarding algorithms for geographical unicast messages and location service mechanism together with all the related data structures (*location service packet buffer* and *forwarding packet buffer*) and extended packet headers.

Is also worth mention that the implementation for the *Security Entity* is soon to start which will allow the GeoNetworking model to support the security mechanisms defined by ETSI in [44] that are missing in the current implementation.

Appendix A

Basic Transport Protocol model

A.1 Introduction

The Basic Transport Protocol layer is implemented as a class called `btp` which is shared between the two facilities layer entities (caBasicService and denBasicService). It has two main functions:

- sendBTP which executes the source operations.
- receiveBTP which executes the destination operations.

A.2 Source Operations

This method gets called by the facilities layer entities when they need to send a new packet. The method call has as argument the `BTP-data.Request` struct with the variables as specified in Appendix C. The C++ code developed to model the BTP source operations is reported below.

```
void
btp::sendBTP(BTPDataRequest_t dataRequest)
{
  GNDataConfirm_t dataConfirm;
  GNDataRequest_t GnDataRequest = {};
  btpHeader header;
  header.SetDestinationPort (dataRequest.destPort);

  if(dataRequest.BTPType==BTP_A) // BTP-A
```

```
    {
      header.SetSourcePort (dataRequest.sourcePort);
      GnDataRequest.upperProtocol = BTP_A;
    }
  else   // BTP-B
    {
      header.SetDestinationPortInfo (dataRequest.destPInfo);
      GnDataRequest.upperProtocol = BTP_B;
    }
  dataRequest.data->AddHeader (header);

  //Filling the GN-dataRequest
  GnDataRequest.GNType = dataRequest.GNType;
  GnDataRequest.GnAddress = dataRequest.GnAddress;
  GnDataRequest.GNCommProfile = dataRequest.GNCommProfile;
  GnDataRequest.GNRepInt = dataRequest.GNRepInt;
  GnDataRequest.GNMaxRepTime = dataRequest.GNMaxRepInt;
  GnDataRequest.GNMaxLife = dataRequest.GNMaxLife;
  GnDataRequest.GNMaxHL = dataRequest.GNMaxHL;
  GnDataRequest.GNTraClass = dataRequest.GNTraClass;
  GnDataRequest.data = dataRequest.data;
  GnDataRequest.lenght = dataRequest.lenght + 4;

  dataConfirm = m_geonet->sendGN(GnDataRequest);
  if(dataConfirm != ACCEPTED)
    {
      NS_LOG_ERROR("GeoNet can't send packet. Error code: " <<
          dataConfirm);
    }

}
```

A.3 Destination Operations

This method gets called by the GN entity when there is a packet that needs to get passed on to the facilities entity. Its argument is the `GNdataIndication` struct with the variables specified in Appendix C. The C++ code developed to model the BTP destination operations is reported below.

```
void
btp::receiveBTP(GNDataIndication_t dataIndication,Address address)
{
 btpHeader header;
```

```
BTPDataIndication_t btpDataIndication = {};
dataIndication.data->RemoveHeader (header, 4);

btpDataIndication.BTPType = dataIndication.upperProtocol;
btpDataIndication.destPort = header.GetDestinationPort ();
if(btpDataIndication.BTPType == BTP_A)
{
  btpDataIndication.sourcePort = header.GetSourcePort ();
  btpDataIndication.destPInfo = 0;
}
else //BTP-B
{
  btpDataIndication.destPInfo = header.GetDestinationPortInfo ();
  btpDataIndication.sourcePort = 0;
}
btpDataIndication.GnAddress = dataIndication.GnAddressDest;
btpDataIndication.GNTraClass = dataIndication.GNTraClass;
btpDataIndication.GNRemPLife = dataIndication.GNRemainingLife;
btpDataIndication.GNPositionV = dataIndication.SourcePV;
btpDataIndication.data = dataIndication.data;
btpDataIndication.lenght = dataIndication.data->GetSize ();

if(btpDataIndication.destPort == CA_PORT)
  m_cam_ReceiveCallback(btpDataIndication,address);
else if(btpDataIndication.destPort == DEN_PORT)
  m_denm_ReceiveCallback(btpDataIndication,address);
else
  NS_LOG_ERROR("BTP : Unknown port");
}
```

Appendix B

GeoNetworking Protocol model

B.1 Network management

As described in Chapter 4 the network management module functionalities from the current implementation include:

- Address configuration
- Local position vector update
- Beaconing

B.1.1 Address configuration

For the current implementation the managed configuration method is adopted as default. At start-up the GeoNet object creates an instance of the GNAddress class which takes the MAC address of the vehicle's ns3 NetDevice i.e., the MID field, and the station type as arguments.

```
m_GNAddress = m_GNAddress.MakeManagedconfiguredAddress
    (m_GnLocalGnAddr,m_stationtype); //! Initial address config on
    MANAGED(1) mode ETSI EN 302 636-4-1 [10.2.1.3]
```

Where the variable `m_GnLocalGnAddr` (i.e., MID field) is set when the device's socket is assigned.

```
void
  GeoNet::setSocketTx(Ptr<Socket> socket_tx)
  {
```

```
    m_socket_tx = socket_tx;
    // ETSI EN 302 636-4-1 [10.2.1.3.2]
    m_GnLocalGnAddr = getGNMac48(m_socket_tx->GetNode ()->GetDevice
        (0)->GetAddress ());
  }
```

Managed address configuration implementation

```
GNAddress GNAddress::MakeManagedconfiguredAddress (Mac48Address addr,
    uint8_t ITSType)
{
  NS_LOG_FUNCTION (addr);
  GNAddress ret;
  uint8_t buf[8];
  uint8_t buf2[8];

  addr.CopyTo (buf);
  //ETSI EN 302 636-4-1 [6.3]
  buf2[0] = 0x00 | ITSType << 2; //Initial GeoNetAddress ->M=0 and
      5bit ITS-S type
  buf2[1] = 0x00; //Reserved
  memcpy (buf2 + 2, buf, 6);

  ret.Set (buf2);
  return ret;
}
```

B.1.2 Ego position vector update

Once the TraCI vehicle data provider (VDP) is assigned to the GeoNet object by the application, a simulator event is schedule to update the EPV after `itsGnMinUpdateFrequencyEPV` [ms]. When the event is triggered GeoNet updates the EPV with the values provided by the VDP and the timestamp is set as the number of elapsed milliseconds since 2004-01-01 00:00:00.000 UTC.

```
GeoNet::EPVupdate ()
{
 if(!(m_stationtype==StationType_roadSideUnit))
 {
   m_egoPV.S_EPV = m_vdp.getSpeedValue ();
   m_egoPV.H_EPV = m_vdp.getHeadingValue ();
   m_egoPV.POS_EPV = m_vdp.getPosition();
```

```
    m_egoPV.TST_EPV = compute_timestampIts (true);

  }
  // ETSI EN 302 636-4-1 [10.2.2]
  //Schedule the next egoPV update with GNMinUpdateFrequencyEPV
     protocol constant (1000ms)
  m_event_EPVupdate = Simulator::Schedule(MilliSeconds
     (m_GnMinUpdateFrequencyEPV),&GeoNet::EPVupdate,this); //! ETSI EN
     302 636-4-1 [10.2.2.2]
 }
```

B.1.3 Beaconing

At startup a GeoAdhoc router sends an initial beacon to announce its presence to other GeoAdhoc routers. This is implemented with the *setBeacon* function which creates a `GN-data.Request` to be sent as shown below.

```
void
GeoNet::setBeacon ()
{
  //Setting GNdataRequest for beacon
  GNDataRequest_t dataRequest = {};
  dataRequest.upperProtocol = ANY_UP;
  dataRequest.GNType = BEACON;
  dataRequest.GNCommProfile = UNSPECIFIED;
  dataRequest.GNMaxLife = 1;
  dataRequest.GNMaxHL = 1;
  dataRequest.GNTraClass = 0x02; // Store carry foward: no - Channel
     offload: no - Traffic Class ID: 2
  dataRequest.lenght = 0;
  Ptr<Packet> packet = Create<Packet>();
  dataRequest.data = packet;
  sendGN (dataRequest);
}
```

B.2 Packet handling

As described in Chapter 4 the Packet handling functionalities from the current implementation include:

- *BEACON* packet handling
- *SHB* packet handling (CAM)

- *GBC* packet handling (DENM)

These functionalities are implemented within 2 main functions called *sendGN* and *receiveGN* due to redundancies in the handling of basic and common packet headers of all types of packets. The sendGN function is called by the sendBTP function explained in Section A.2 and takes as argument a `GNdataRequest` while receiveGN is called every time the bonded socket receives a packet.

B.2.1 Source operations

The C++ code developed to model the GN source operations is reported below.

```
 GNDataConfirm_t
GeoNet::sendGN (GNDataRequest_t dataRequest)
{
  GNDataConfirm_t dataConfirm = ACCEPTED;
  GNBasicHeader basicHeader;
  GNCommonHeader commonHeader;
  GNlpv_t longPV;

  if(dataRequest.lenght > m_GnMaxSduSize)
  {
    return MAX_LENGHT_EXCEEDED;
  }
  if(dataRequest.GNMaxLife > m_GNMaxPacketLifetime)
  {
     return MAX_LIFE_EXCEEDED;
  }
  if(dataRequest.GNRepInt != 0)
  {
    if(dataRequest.GNRepInt < m_GNMinPacketRepetitionInterval)
    {
      return REP_INTERVAL_LOW;
    }
  }
```

Basic header

```
//Basic Header field setting according to ETSI EN 302 636-4-1 [10.3.2]
  basicHeader.SetVersion (m_GnPtotocolVersion);
  //Security option not implemented
  basicHeader.SetNextHeader (1);//! Next Header: Common Header (1)
  if(dataRequest.GNMaxLife != 0)
```

```
  {
    basicHeader.SetLifeTime(encodeLT(dataRequest.GNMaxLife));
  }
  else
  {
    basicHeader.SetLifeTime(encodeLT(m_GnDefaultPacketLifetime));
  }
  if(dataRequest.GNMaxHL != 0)
  {
    basicHeader.SetRemainingHL (dataRequest.GNMaxHL);
  }
  else
  {
    basicHeader.SetRemainingHL (m_GnDefaultHopLimit);
  }
```

Common header

```
//Common Header field setting according to ETSI EN 302 636-4-1
    [10.3.4]
commonHeader.SetNextHeader(dataRequest.upperProtocol); //0 if
    ANY(beacon) 1 if btp-a or 2 if btp-b
commonHeader.SetHeaderType(dataRequest.GNType);

commonHeader.SetHeaderSubType(0); //By default set header sub-type to
    0
if((dataRequest.GNType == GBC) || (dataRequest.GNType == GAC) )
  commonHeader.SetHeaderSubType(dataRequest.GnAddress.shape); //If GBC
      or GAC specify the shape of the GeoArea addressed
else if(dataRequest.GNType == TSB)
  {
    if(dataRequest.GNMaxHL>1)
      commonHeader.SetHeaderSubType(1);//For now shouldn't happen
  }



commonHeader.SetTrafficClass (dataRequest.GNTraClass);
commonHeader.SetFlag(m_GnIsMobile);

if(dataRequest.GNMaxHL != 0)// Equal to the remaining hop limit
{
  commonHeader.SetMaxHL (dataRequest.GNMaxHL);
```

```
}
else
{
  commonHeader.SetMaxHL(m_GnDefaultHopLimit);
}

commonHeader.SetPayload (dataRequest.lenght);
```

Long position vector

```
  longPV.GnAddress = m_GNAddress;
  longPV.TST = m_egoPV.TST_EPV;
  longPV.latitude = (int32_t) (m_egoPV.POS_EPV.y*DOT_ONE_MICRO);
  longPV.longitude = (int32_t) (m_egoPV.POS_EPV.x*DOT_ONE_MICRO);
  longPV.positionAccuracy = false;
  longPV.speed = (int16_t) m_egoPV.S_EPV;
  longPV.heading = (uint16_t) m_egoPV.H_EPV;
```

Extended Header

```
switch(dataRequest.GNType)
  {
  case BEACON:
    if(commonHeader.GetHeaderSubType ()==0) dataConfirm = sendBeacon
        (dataRequest,commonHeader,basicHeader,longPV);
    break;
  case GBC:
    if(commonHeader.GetHeaderSubType ()==0) dataConfirm = sendGBC
        (dataRequest,commonHeader,basicHeader,longPV);
    break;
  case TSB:
    if(commonHeader.GetHeaderSubType ()==0) dataConfirm = sendSHB
        (dataRequest,commonHeader,basicHeader,longPV);
    break;
  default:
    NS_LOG_ERROR("GeoNet packet not supported");
    dataConfirm = UNSPECIFIED_ERROR;
  }
  return dataConfirm;
}
```

BEACON packet

If the Packet Transport Type parameter in the GNdataRequest is set to 1, the setting of the Beacon packet starts as reported below.

```
GNDataConfirm_t
GeoNet::sendBeacon(GNDataRequest_t dataRequest,GNCommonHeader
    commonHeader, GNBasicHeader basicHeader,GNlpv_t longPV)
{
  BeaconHeader header;
  //ETSI EN 302 636-4-1 [10.3.6]
  //1)Create GN-PDU, steps a) and b) already done in sendGN()
  //c) set the fields of the beacon extended header
  header.SetLongPositionV (longPV);
  dataRequest.data->AddHeader (header);

  dataRequest.data->AddHeader (commonHeader);
  dataRequest.data->AddHeader (basicHeader);
  //!2)Security profile settings not implemented
  //!3)Media-dependent procedures not implemented
  //4) pass the GN-PDU to LL protocol entity
  if(m_socket_tx==NULL)
  {
    NS_LOG_ERROR("GeoNet: SOCKET NOT FOUND ");
    return UNSPECIFIED_ERROR;
  }
  //if(!m_GnIsMobile)return ACCEPTED;
  if(m_socket_tx->Send (dataRequest.data)==-1)
  {
    NS_LOG_ERROR("Cannot send BEACON packet ");
    return UNSPECIFIED_ERROR;
  }
  m_event_Beacon.Cancel ();
  //5) initialize timer for periodic retransmission of beacons
  double T_beacon = m_GnBeaconServiceRetransmitTimer + (rand()%
      m_GnBeaconServiceMaxJItter);
  m_event_Beacon =
      Simulator::Schedule(MilliSeconds(T_beacon),&GeoNet::setBeacon,this);
  return ACCEPTED;
}
```

SHB packet

If the Packet Transport Type parameter in the `GN-data.Request` is set to 5, the setting of the SHB header starts as reported below.

```
GNDataConfirm_t
GeoNet::sendSHB (GNDataRequest_t dataRequest,GNCommonHeader
    commonHeader, GNBasicHeader basicHeader,GNlpv_t longPV)
{
  SHBheader header;
  //1) Create SHB GN-PDU with SHB header setting according to ETSI EN
     302 636-4-1 [10.3.10.2]
  //a) and b) already done
  //c) SHB extended header
  header.SetLongPositionV (longPV);
  dataRequest.data->AddHeader (header);

  dataRequest.data->AddHeader (commonHeader);
  dataRequest.data->AddHeader (basicHeader);

  //2)Security setting -not implemeted yet-
  //3)If not suitable neighbour exist in the LocT and the SCF for the
     traffic class is set:

  if((dataRequest.GNTraClass > 128) && (!hasNeighbour ()))
  {
    //a)Buffer the SHB packet in the BC forwarding buffer and omit
       execution of further steps
    return UNSUPPORTED_TRA_CLASS;//Not implemented yet
  }
  //4)If the optional repetition interval paramter in the
     GN-dataRequest parameter is set
  if(dataRequest.GNRepInt != 0)
  {
    if(dataRequest.GNRepInt < m_GNMinPacketRepetitionInterval) return
       REP_INTERVAL_LOW;
    //a)save the SHB packet
    saveRepPacket (dataRequest);
  }
  //5)Media dependent procedures -Omited-
  //6)Pass the GN-PDU to the LL protocol entity
  if(m_socket_tx==NULL)
  {
    NS_LOG_ERROR("GeoNet: SOCKET NOT FOUND ");
    return UNSPECIFIED_ERROR;
```

```
  }
  if(m_socket_tx->Send (dataRequest.data)==-1)
    {
      NS_LOG_ERROR("Cannot send SHB packet ");
      return UNSPECIFIED_ERROR;
    }
  //7)reset beacon timer to prevent dissemination of unnecessary
      beacon packet
  m_event_Beacon.Cancel ();
  double T_beacon = m_GnBeaconServiceRetransmitTimer + (rand()%
      m_GnBeaconServiceMaxJItter);
  m_event_Beacon =
      Simulator::Schedule(MilliSeconds(T_beacon),&GeoNet::setBeacon,this);
  return ACCEPTED;
}
```

GBC packet

If the Packet Transport Type parameter in the `GN-data.Request` is set to 4, the setting of the GBC header starts as reported below.

```
GNDataConfirm_t
GeoNet::sendGBC (GNDataRequest_t dataRequest,GNCommonHeader
    commonHeader, GNBasicHeader basicHeader,GNlpv_t longPV)
{
  GBCheader header;
  //1) Create SHB GN-PDU with GBC header setting according to ETSI EN
      302 636-4-1 [10.3.11.2]
  //a) and b) already done
  //GBC extended header
  header.SetSeqNumber (m_seqNumber);
  header.SetLongPositionV (longPV);
  header.SetGeoArea(dataRequest.GnAddress);

  dataRequest.data->AddHeader (header);

  dataRequest.data->AddHeader (commonHeader);
  dataRequest.data->AddHeader (basicHeader);
  /*
   * 2)If not suitable neighbour exist in the LocT and the SCF for the
       traffic class is set:
   * a)Buffer the SHB packet in the BC forwarding buffer and omit
       execution of further steps
  */
```

```
if((dataRequest.GNTraClass >= 128) && (!hasNeighbour ()))
{
  return UNSUPPORTED_TRA_CLASS;//Not implemented yet
}
//!3)Execute forwarding algorithm selection procedure, not
    implemented because RSU in our case will always be inside the
    target area
//4) if packet is buffered in any of the forwarding packets, omit
    further steps
if(m_GNAreaForwardingAlgorithm == 2)
{
  /* push packet into CBF buffer */
  return UNSPECIFIED_ERROR;
}
//!5)Security profile settings not implemented
//6)If the optional repetition interval paramter in the
    GN-dataRequest parameter is set
if(dataRequest.GNRepInt != 0)
{
  if(dataRequest.GNRepInt < m_GNMinPacketRepetitionInterval) return
     REP_INTERVAL_LOW;
  //a)save the SHB packet
  saveRepPacket(dataRequest);
}
//!7)Media dependent procedures -Omited-
//8)Pass the GN-PDU to the LL protocol entity
if(m_socket_tx==NULL)
{
  NS_LOG_ERROR("GeoNet: SOCKET NOT FOUND ");
  return UNSPECIFIED_ERROR;
}

if(m_socket_tx->Send (dataRequest.data)==-1)
{
  NS_LOG_ERROR("Cannot send GBC packet ");
  return UNSPECIFIED_ERROR;
}

//*reset beacon timer to prevent dissemination of unnecessary beacon
    packet
m_event_Beacon.Cancel ();
double T_beacon = m_GnBeaconServiceRetransmitTimer + (rand()%
    m_GnBeaconServiceMaxJItter);
m_event_Beacon =
    Simulator::Schedule(MilliSeconds(T_beacon),&GeoNet::setBeacon,this);
```

```
  //Update sequence number
  m_seqNumber = (m_seqNumber+1)% SN_MAX;
  return ACCEPTED;
}
```

B.2.2 Destination operations

The C++ code developed to model the GN destination operations is reported below.

```
void
GeoNet::receiveGN(Ptr<Socket> socket)
{
  GNDataIndication_t dataIndication = {};
  Address from;
  GNBasicHeader basicHeader;
  GNCommonHeader commonHeader;

  dataIndication.data = socket->RecvFrom (from);

  dataIndication.data->RemoveHeader (basicHeader, 4);
  dataIndication.GNRemainingLife = basicHeader.GetLifeTime ();
  dataIndication.GNRemainingHL = basicHeader.GetRemainingHL ();
```

Basic header

The Basic Header processing consist in checking that the version field matches the GeoAdhoc Router's own version and checking which is the next header. In the current implementation the next header is always the Common Header. The *Remaining Packet Lifetime* and *Remaining Hop Limit* are passed to the `GN-data.Indication`.

```
//Basic Header Procesing according to ETSI EN 302 636-4-1 [10.3.3]
  //1)Check version field
  if(basicHeader.GetVersion() != m_GnPtotocolVersion)
  {
    NS_LOG_ERROR("Incorrect version of GN protocol");
  }
  //2)Check NH field
  if(basicHeader.GetNextHeader()==2) //a) if NH=0 or NH=1 proceed with
      common header procesing
  {
    //Secured packet
```

```
  }
  dataIndication.GNRemainingLife = decodeLT(basicHeader.GetLifeTime
      ());
```

Common header

```
//Common Header Processing according to ETSI EN 302 636-4-1 [10.3.5]
 dataIndication.data->RemoveHeader (commonHeader, 8);
 dataIndication.upperProtocol = commonHeader.GetNextHeader ();
    //!Information needed for step 7
 dataIndication.GNTraClass = commonHeader.GetTrafficClass ();
    //!Information needed for step 7
 //1) Check MHL field
 if(commonHeader.GetMaxHopLimit() < basicHeader.GetRemainingHL())
 {
   NS_LOG_ERROR("Max hop limit greater than remaining hop limit");
      //a) if MHL<RHL discard packet and omit execution of further
      steps
   return;
 }
 //2) process the BC forwarding buffer, for now not implemented (SCF
    in traffic class disabled)
 //3) check HT field
 dataIndication.GNType = commonHeader.GetHeaderType();
 dataIndication.lenght = commonHeader.GetPayload ();
```

Extended Header

```
switch(dataIndication.GNType)
   {
     case BEACON:
       if(commonHeader.GetHeaderSubType ()==0) processSHB
           (dataIndication,from);
       break;
     case GBC:
       if(commonHeader.GetHeaderSubType ()==0) processGBC
           (dataIndication,from);
       break;
     case TSB:
       if((commonHeader.GetHeaderSubType ()==0)) processSHB
           (dataIndication,from);
       break;
```

```
    default:
      NS_LOG_ERROR("GeoNet packet not supported");
   }
 }
```

SHB packet / Beacon packet

After processing the Basic and Common header if the Header Type field in the latter is equal to 1 or 5 the SHB/Beacon processing gets started as reported below.

```
void
GeoNet::processSHB (GNDataIndication_t dataIndication, Address from)
{
  SHBheader shbHeader;
  BeaconHeader beaconHeader;
  if(dataIndication.GNType == BEACON)
  {
    dataIndication.data->RemoveHeader (beaconHeader, 24);
    dataIndication.SourcePV = beaconHeader.GetLongPositionV ();
  }
  else
  {
    dataIndication.data->RemoveHeader (shbHeader, 28);
    dataIndication.SourcePV = shbHeader.GetLongPositionV ();
  }
  // SHB Processing according to ETSI EN 302 636-4-1 [10.3.10.3] or
     Beacon processing according to [10.3.6.3]
  //3)execute DAD
  if(m_GnLocalAddrCongMethod == 0)
  {
    if(DAD(dataIndication.SourcePV.GnAddress))
    {
       NS_LOG_ERROR("Duplicate address detected");
    }
  }
  //4)update PV in the SO LocTE with the SO PV fields of the SHB
     extended header
  m_LocT_Mutex.lock ();
  std::map<GNAddress, GNLocTE>::iterator entry_map_it =
     m_GNLocT.find(dataIndication.SourcePV.GnAddress);

  //Not specified in the protocol but first check if LocTE exist in
     the LocTable
  if (entry_map_it == m_GNLocT.end())
```

```
  {
    newLocTE (dataIndication.SourcePV);
  }
  else
  {
    //Update LongPV
    LocTUpdate (dataIndication.SourcePV,entry_map_it);
    //6)Set IS_NEIGHBOUR flag to true
    entry_map_it->second.IS_NEIGHBOUR = true;
  }
  m_LocT_Mutex.unlock ();
  //7) Pass the payload to the upper protocol entity if it's not a
      beacon packet
  if(dataIndication.GNType != BEACON)
  {
    m_ReceiveCallback(dataIndication,from);
  }
}
```

GBC packet

After processing the Basic and Common header if the Header Type field in the latter is equal to 1 or 5 the GBC processing gets started as reported below.

```
void
GeoNet::processGBC (GNDataIndication_t dataIndication,Address from)
{
  // GBC Processing according to ETSI EN 302 636-4-1 [10.3.11.3] 1 and
      2 already done in receiveGN method
  GBCheader header;
  dataIndication.data->RemoveHeader (header,56);
  dataIndication.SourcePV = header.GetLongPositionV ();
  dataIndication.GnAddressDest = header.GetGeoArea ();
  dataIndication.GnAddressDest.shape = shape;

  //3)Determine function F as specified in ETSI EN 302 931
  m_LocT_Mutex.lock ();
  std::map<GNAddress, GNLocTE>::iterator entry_map_it =
      m_GNLocT.find(dataIndication.SourcePV.GnAddress);
  if(entry_map_it != m_GNLocT.end())
  {
    //a)
    if((!isInsideGeoArea (dataIndication.GnAddressDest)) &&
        ((m_GNNonAreaForwardingAlgorithm==0)||(m_GNNonAreaForwardingAlgorithm==1)))
```

```
    {
      //execute DPD as specified in A.2
      if(DPD(header.GetSeqNumber (),dataIndication.SourcePV.GnAddress))
      {
        NS_LOG_ERROR("Duplicate received");
        return;
      }
    }
    //b)
    if((isInsideGeoArea (dataIndication.GnAddressDest)) &&
        ((m_GNAreaForwardingAlgorithm==0)||(m_GNAreaForwardingAlgorithm==1)))
    {
      //execute DPD as specified in A.2
      if(DPD(header.GetSeqNumber (),dataIndication.SourcePV.GnAddress))
      {
        NS_LOG_ERROR("Duplicate received");
        return;
      }
    }
  }
  //4) DAD
  if (m_GnLocalAddrCongMethod == 0)
  {
    if(DAD(dataIndication.SourcePV.GnAddress))
    {
       NS_LOG_ERROR("Duplicate address detected");
    }
  }
  //Check for LocTE existence
  if (entry_map_it == m_GNLocT.end())
  {
    //5) If LocTE doesn't exist
    newLocTE (dataIndication.SourcePV);//a) create PV with the SO PV
        in the extended header
    m_GNLocT[dataIndication.SourcePV.GnAddress].IS_NEIGHBOUR =
        false;//b) Set the IS_NEIGHBOUR flag to FALSE
    //c) PDR not implemented yet
  }
  else
  {
    //6)If the LocTe exist update LongPV, PDR not implemented yet
    LocTUpdate (dataIndication.SourcePV,entry_map_it);
  }
  m_LocT_Mutex.unlock ();
  //7)Determine function F(x,y) as specified in ETSI EN 302 931
```

```
  if(isInsideGeoArea (dataIndication.GnAddressDest))
  {
    //a) Pass the payload to the upper protocol entity
    dataIndication.GNType = GBC;
    dataIndication.lenght = dataIndication.data->GetSize ();
    m_ReceiveCallback(dataIndication,from);
  }
  else
  {
    NS_LOG_INFO("GBC packet discarded because GeoNetworking reported
        it as out-of-range");
  }
}
```

B.2.3 Geographical Area

```
typedef struct _geoarea {
  int32_t posLong; // Longitude
  int32_t posLat; // Latitude
  uint16_t distA; // Distance a
  uint16_t distB; // Distance b
  uint16_t angle; // Angle
}GeoArea_t;
```

Determining if vehicle is inside Geographical Area

The C++ code for the function used to check if the GeoAdhoc Router is inside the GeoArea extracted from the GBC header or not, is reported below.

```
bool
 GeoNet::isInsideGeoArea (GeoArea_t geoArea)
 {
   //Compute function F specified in ETSI EN 302 931
   double x,y,r,f,geoLon,geoLat;
   VDP::VDP_position_cartesian_t egoPos, geoPos;
   if((m_egoPV.POS_EPV.lat==0)||(m_egoPV.POS_EPV.lon==0))return
       false;//In case egoPV hasnt updated for the first time yet

   egoPos = m_vdp->getXY(m_egoPV.POS_EPV.lon,m_egoPV.POS_EPV.lat);
       //Compute cartesian position of the vehicle
   geoLon = ((double) geoArea.posLong)/DOT_ONE_MICRO;
   geoLat = ((double)geoArea.posLat)/DOT_ONE_MICRO;
```

```
geoPos = m_vdp->getXY(geoLon, geoLat); // Compute cartesian position
    of the geoArea center

/*(x,y) is the cartesian position relative to the center of the
    geoArea shape -> if vehicle is indeed in the center of
 * the shape, (x,y)=(0,0)
*/
x = egoPos.x - geoPos.x;
y = egoPos.y - geoPos.y;

if(geoArea.shape == 0)
  {
    //Circular area
    r = geoArea.distA;
    f = 1.0 - pow((x/r),2.0) - pow((y/r),2.0);
  }
else
  {
    //Rotate (x,y) clockwise around the center of the shape by the
        zenith angle = 90 deg - azimuth
    double xr,yr,zenith;
    zenith = (90.0 - geoArea.angle)* (M_PI/180.0); // zenith in
        radians
    xr = x * cos(zenith) + y * sin(zenith);
    yr = -(x * sin(zenith)) + y * cos(zenith);
    if (geoArea.shape == 1)
    {
      //Rectangular area function ETSI EN 302 931
      f = std::min(1.0 - pow(xr/geoArea.distA,2.0), 1.0 -
          pow(yr/geoArea.distB,2.0));
    }
    else
      {
        //Ellipsoidal area
        f = 1.0 - pow((xr/geoArea.distA),2.0) -
            pow((yr/geoArea.distB),2.0);
      }
  }

if(f>=0)
{
  return true;// Local router inside the geographical target
}
else
{
```

```
    return false; // Local router outside the geographical target
  }
}
```

B.2.4 Duplicate Address Detection

The C++ code for the Duplicate Address Detection (DAD) implementation, is reported below. The function returns `true` if a duplicate address is detected or `false` otherwise.

```
bool
  GeoNet::DAD(GNAddress address)
  {
    if((address == m_GNAddress) && (address.GetLLAddress
        ()==m_GNAddress.GetLLAddress ()))
    {
      //ETSI EN 302 636-4-1 [10.2.1.5] : If conflict is detected,
         request new MID field
      m_GnLocalGnAddr = getGNMac48(m_socket_tx->GetNode ()->GetDevice
          (0)->GetAddress ());
      m_GNAddress = m_GNAddress.MakeManagedconfiguredAddress
          (m_GnLocalGnAddr,m_stationtype);
      return true;
    }
    else
    {
      return false;
    }
  }
```

B.2.5 Duplicate Packet Detection

The C++ code for the Duplicate Packet Detection (DPD) implementation, is reported below. The function returns `true` if a duplicate packet is detected or `false` otherwise.

```
bool
GeoNet::DPD(uint16_t seqNumber,GNAddress address)
{
  std::set<uint16_t>::iterator it =
     m_GNLocT[address].DPL.find(seqNumber);
  if(it == m_GNLocT[address].DPL.end ())
  {
```

```
    //If entry doesnt exist, the packet is not a duplicate and should be
        added to the list
    m_GNLocT[address].DPL.insert (seqNumber);
    return false;
  }
  else
  {
    return true;//Packet is a duplicate
  }
}
```

B.2.6 Location Table handling

New location table entry creation

The C++ code for the creation of a new Location Table Entry (LocTE), as described in Chapter 4, is reported below.

```
void
GeoNet::newLocTE (GNlpv_t lpv)
{
  //Create new LocT entry according to ETSI EN 302 636-4-1 [8.1.2]
  GNLocTE new_entry;
  new_entry.GN_ADDR = lpv.GnAddress;
  new_entry.LL_ADDR = lpv.GnAddress.GetLLAddress();
  new_entry.version = 1;
  new_entry.lpv = lpv;
  new_entry.LS_PENDING = false;
  new_entry.IS_NEIGHBOUR = true;
  new_entry.DPL.clear ();//!Duplicate packet list
  new_entry.timestamp = compute_timestampIts (true);
  new_entry.PDR = 0; //!Packet data rate, yet to be implemented

  //Before storing the new entry, start the T(LocTE) as specified in
      [8.1.3]
  m_GNLocTTimer.emplace(lpv.GnAddress, Timer());
  GeoNet::setTLocT(m_GNLocTTimer[lpv.GnAddress],Seconds(m_GnLifeTimeLocTE),&GeoNet::L
  //!LS_PENDING timer [8.1.3] not implemented yet

  //Store new entry
  m_GNLocT.emplace(lpv.GnAddress,new_entry);
}
```

Location table entry update

The C++ code for the update of a Location Table Entry (LocTE), as described in Chapter 4, is reported below.

```
void
  GeoNet::LocTUpdate (GNlpv_t lpv,
      std::map<GNAddress,GNLocTE>::iterator locte_it)
  {
    //Update LongPV as especified in clause C.2
    long TSTpv_rp = lpv.TST; //! Timestamp for the position vector in
        the received GN packet
    long TSTpv_locT = locte_it->second.timestamp; //! Timestamp for the
        position vector in the LocT to be updated
    if(((TSTpv_rp>TSTpv_locT) && ((TSTpv_rp-TSTpv_locT)<=TS_MAX/2)) ||
       ((TSTpv_locT>TSTpv_rp) && ((TSTpv_locT-TSTpv_rp)>TS_MAX/2)))
    {
      //TSTpv_rp greater than TSTpv_locT
      //Cancel T(LocTE) from previous entry even if this is done by
          setTLocT
      m_GNLocTTimer[lpv.GnAddress].Cancel();
      //Before updating entry, start the T(LocTE) as specified in [8.1.3]
      m_GNLocTTimer.emplace(lpv.GnAddress, Timer ());
      GeoNet::setTLocT(m_GNLocTTimer[lpv.GnAddress],Seconds(m_GnLifeTimeLocTE),&GeoNet
      //PVlocT <- PVrp
      locte_it->second.lpv = lpv;
    }
    else
    {
      NS_LOG_ERROR("Timestamp of received packet is not greater than the
          one from LocT !");
    }
  }
```

Appendix C

Service Primitives

C.1 BTP service primitives

The BTP data service primitives allow entities of ITS Facilities protocols to send and receive PDUs via the BTP_SAP [35]. There are two:

- BTP-Data.request.
- BTP-Data.indication.

C.1.1 BTP-Data.request

The BTP-Data.request primitive is used by the ITS Facilities protocol entity to request sending a BTP-PDU [35]. In MS-VAN3T the BTP-Data.request is implemented with the following variables:

```
typedef struct _btpdataRequest {
    BTPType_t BTPType;
    int16_t destPort;
    int16_t sourcePort;
    int16_t destPInfo;

    TransportType_t GNType; // GN Packet transport type -- GeoUnicast,
        SHB, TSB, GeoBroadcast or GeoAnycast
    GeoArea_t GnAddress; // GN destination adress -- destination
        adress(GeoUnicast) or geo. area (GeoBroadcast or GeoAnycast)
    CommProfile_t GNCommProfile; // GN Communication Profile --
        determines de LL protocol entity

    int16_t GNSecurityP; // GN Security Profile /OPCIONAL/
    double GNMaxLife; //GN Maximum Packet Lifetime in [s] /OPCIONAL/
```

```
    int16_t GNRepInt; // GN Repetition Interval /OPCIONAL/
    int16_t GNMaxRepInt; // GN maximum repetition Interval /OPCIONAL/
    int16_t GNMaxHL; // GN Max Hop Limit /OPCIONAL/
    uint8_t GNTraClass; // GN Traffic Class

    uint32_t lenght; // Payload size
    Ptr<Packet> data; // Payload
  } BTPDataRequest_t;
```

C.1.2 BTP-Data.indication

The BTP-Data.indication primitive indicates to an ITS facilities layer protocol entity that a ITS-FSDU has been received [35]. In MS-VAN3T the BTP-Data.indication is implemented with the following variables:

```
typedef struct _dataIndication {
    uint8_t BTPType;
    int16_t destPort;
    int16_t sourcePort;
    int16_t destPInfo;

    uint8_t GNType; // GN Packet transport type -- GeoUnicast, SHB, TSB,
       GeoBroadcast or GeoAnycast
    GeoArea_t GnAddress; // GN destination adress -- destination
       adress(GeoUnicast) or geo. area (GeoBroadcast or GeoAnycast)
    GNlpv_t GNPositionV; // GN Posistion vector

    int16_t GNSecurityR; // GN Security Report /OPCIONAL/
    int16_t GNCertID; //GN Certificate ID /OPCIONAL/
    int16_t GNPermissions; // GN Permissions /OPCIONAL/
    int16_t GNMaxRepInt; // GN maximum repetition Interval /OPCIONAL/
    uint8_t GNTraClass; // GN Traffic Class
    double GNRemPLife; // GN Reamianing Packet Lifetime /OPCIONAL/

    uint32_t lenght;
    Ptr<Packet> data;
  } BTPDataIndication_t;
```

C.2 GeoNetworking service primitives

The GN data service primitives allow entities of ITS transport protocols to send and receive PDUs via the GN_SAP. There are three:

- GN-Data.request.
- GN-Data.indication.
- GN-Data.confirm.

C.2.1 GN-Data.request

The service primitive GN-Data.request is used by the ITS transport protocol entity to request sending a GeoNetworking packet [34]. In MS-VAN3T the GN-Data.request is implemented with the following variables:

```
typedef struct _gndataRequest {
    BTPType_t upperProtocol;
    TransportType_t GNType; // GN Packet transport type -- GeoUnicast,
        SHB, TSB, GeoBroadcast or GeoAnycast
    GeoArea_t GnAddress; // GN destination adress -- destination
        address(GeoUnicast) or geo. area (GeoBroadcast or GeoAnycast)
    CommProfile_t GNCommProfile; // GN Communication Profile --
        determines de LL protocol entity

    int16_t GNSecurityP; // GN Security Profile /OPCIONAL/
    int32_t GNITS_AIDL;//Length of the value provided in the ITS-AID
        parameter /OPCIONAL/
    int32_t GNITS_AID;// ITS-AID for the payload to be sent/OPCIONAL/
    int32_t GNSecurityPermL;// Length of the security permissions
        parameter/OPCIONAL/
    int32_t GNSecurityPerm;// SSP associated with the ITS-AID/OPCIONAL/
    int32_t GNSecurityContInfo;// Information to be used to selecting
        properties of the security protocol/OPCIONAL/
    int32_t GNSecurityTargetIDListL; //Length for the value in the
        security target id list parameter /OPCIONAL/
    int32_t GNSecurityTargetIDList;// Unordered collection of target IDs
        used by the security entity for specifying multiple recipients
        /OPCIONAL/
    double GNMaxLife; //GN Maximum Packet Lifetime in [s] /OPCIONAL/
    int16_t GNRepInt; // GN Repetition Interval /OPCIONAL/
    int16_t GNMaxRepTime; // GN maximum repetition time /OPCIONAL/
    int16_t GNMaxHL; // GN Max Hop Limit /OPCIONAL/
    uint8_t GNTraClass; // GN Traffic Class

    uint32_t lenght; // Payload size
    Ptr<Packet> data; // Payload

    bool operator <( const _gndataRequest &rhs ) const
```

```
        {
            return ( data < rhs.data );
        }
 } GNDataRequest_t;
```

C.2.2 GN-Data.indication

The service primitive GN-Data.indication indicates to an upper protocol entity that a GeoNetworking packet has been received [34]. In MS-VAN3T the GN-Data.indication is implemented with the following variables:

```
typedef struct _gndataIndication {
    uint8_t upperProtocol;
    uint8_t GNType; // GN Packet transport type -- GeoUnicast, SHB, TSB,
        GeoBroadcast or GeoAnycast
    GeoArea_t GnAddressDest; // GN destination adress -- destination
        adress(GeoUnicast) or geo. area (GeoBroadcast or GeoAnycast)
        with which the packet was generated with
    GNlpv_t SourcePV;
    //Security report /OPCIONAL/
    //Certificate ID/OPCIONAL/
    int32_t GNITS_AIDL;//Length of the value provided in the ITS-AID
        parameter /OPCIONAL/
    int32_t GNITS_AID;// ITS-AID for the payload to be sent/OPCIONAL/
    int32_t GNSecurityPermL;// Length of the security permissions
        parameter/OPCIONAL/
    int32_t GNSecurityPerm;// SSP associated with the ITS-AID/OPCIONAL/
    int32_t GNSecurityContInfo;// Information to be used to selecting
        properties of the security protocol/OPCIONAL/

    uint8_t GNTraClass; // GN Traffic Class
    double GNRemainingLife; //GN Remaining Packet Lifetime in [s]
        /OPCIONAL/
    int16_t GNRemainingHL; // GN Remaining Hop Limit /OPCIONAL/

    uint32_t lenght; // Payload size
    Ptr<Packet> data; // Payload
 } GNDataIndication_t;
```

C.2.3 GN-Data.confirm

The service primitive GN-Data.confirm is used to confirm that the GeoNetworking packet was successfully processed in response to a GN-Data.request [34]. In MS-VAN3T the GN-Data.confirm is used as return value of the function sendGN with the following return values:

```
typedef enum {
   ACCEPTED=0,
   MAX_LENGHT_EXCEEDED = 1,
   MAX_LIFE_EXCEEDED = 2,
   REP_INTERVAL_LOW = 3,
   UNSUPPORTED_TRA_CLASS = 4,
   MAX_GEOAREA_EXCEEDED = 5,
   UNSPECIFIED_ERROR =6
 }GNDataConfirm_t;
```

Bibliography

[1] M. Malinverno, G. Avino, C. Casetti, C. F. Chiasserini, F. Malandrino, and S. Scarpina. «Edge-Based Collision Avoidance for Vehicles and Vulnerable Users: An Architecture Based on MEC». In: *IEEE Vehicular Technology Magazine* 15.1 (2020), pp. 27–35 (cit. on p. 2).

[2] G Avino, M Malinverno, C Casetti, C. F Chiasserini, F Malandrino, M Rapelli, and G Zennaro. «Support of Safety Services through Vehicular Communications: The Intersection Collision Avoidance Use Case». eng. In: IEEE, 2018 (cit. on p. 2).

[3] José Víctor Saiáns-Vázquez, Esteban Fernando Ordóñez-Morales, Martín López-Nores, Yolanda Blanco-Fernández, Jack Fernando Bravo-Torres, José Juan Pazos-Arias, Alberto Gil-Solla, and Manuel Ramos-Cabrer. «Intersection Intelligence: Supporting Urban Platooning with Virtual Traffic Lights over Virtualized Intersection-Based Routing». In: *Sensors* 18.11 (2018). ISSN: 1424-8220. DOI: 10.3390/s18114054. URL: https://www.mdpi.com/1424-8220/18/11/4054 (cit. on p. 2).

[4] Michel Ferreira, Ricardo Fernandes, Hugo Conceição, Wantanee Viriyasitavat, and Ozan K. Tonguz. «Self-Organized Traffic Control». In: *Proceedings of the Seventh ACM International Workshop on VehiculAr InterNETworking*. VANET '10. Chicago, Illinois, USA: Association for Computing Machinery, 2010, pp. 85–90. ISBN: 9781450301459. DOI: 10.1145/1860058.1860077. URL: https://doi.org/10.1145/1860058.1860077 (cit. on p. 2).

[5] A. A. Alam, A. Gattami, and K. H. Johansson. «An experimental study on the fuel reduction potential of heavy duty vehicle platooning». In: *13th International IEEE Conference on Intelligent Transportation Systems*. 2010, pp. 306–311. DOI: 10.1109/ITSC.2010.5625054 (cit. on p. 3).

[6] H. Y. Chang, H. W. Lin, Z. H. Hong, and T. L. Lin. «A Novel Algorithm for Searching Parking Space in Vehicle Ad Hoc Networks». In: *2014 Tenth International Conference on Intelligent Information Hiding and Multimedia Signal Processing*. 2014, pp. 686–689. DOI: 10.1109/IIH-MSP.2014.177 (cit. on p. 3).

[7] Sherali Zeadally et al. «Vehicular ad hoc networks (VANETS): status, results, and challenges». eng. In: *Telecommunication systems* 50.4 (2012), pp. 217–241. ISSN: 1018-4864 (cit. on pp. 3, 7).

[8] Elyes Ben Hamida, Hassan Noura, and Wassim Znaidi. «Security of Cooperative Intelligent Transport Systems: Standards, Threats Analysis and Cryptographic Countermeasures». In: *Electronics* 4.3 (2015), pp. 380–423. ISSN: 2079-9292. DOI: 10.3390/electronics4030380. URL: https://www.mdpi.com/2079-9292/4/3/380 (cit. on p. 4).

[9] Hannes Hartenstein and Kenneth Laberteaux. *VANET: Vehicular Applications and Inter-Networking Technologies*. Jan. 2009, pp. 1–435. DOI: 10.1002/9780470740637 (cit. on p. 4).

[10] *FCC reallocates transportation safety spectrum for Wi-Fi use, endorses C-V2X for auto safety*. URL: https://www.engage.hoganlovells.com/knowledgeservices/news/fcc-reallocates-transportation-safety-spectrum-for-wi-fi-use-endorses-c-v2x-for-auto-safety (cit. on p. 5).

[11] *8/671/EC- Commission Decision of 5 August 2008 on the Harmonised use of Radio Spectrum in the 5875-5905 MHz Frequency Band for Safety-Related Applications of Intelligent Transport Systems (ITS)*. Standard. European Commission, 2008 (cit. on p. 5).

[12] J. Choi, V. Marojevic, C. B. Dietrich, J. H. Reed, and S. Ahn. «Survey of Spectrum Regulation for Intelligent Transportation Systems». In: *IEEE Access* 8 (2020), pp. 140145–140160. DOI: 10.1109/ACCESS.2020.3012788 (cit. on p. 5).

[13] *ETSI TR 102 492-1 V1.1.1 - Electromagnetic compatibility and Radio spectrum Matters (ERM); Intelligent Transport Systems (ITS); Part 1: Technical characteristics for pan-European harmonized communications equipment operating in the 5 GHz frequency range and intended for critical road-safety applications; System Reference Document*. Technical Requirement. European Telecommunication Standard Institute, 2006 (cit. on p. 5).

[14] *ETSI TR 102 492-2 V1.1.1 - Electromagnetic compatibility and Radio spectrum Matters (ERM); Intelligent Transport Systems (ITS); Part 2: Technical characteristics for pan European harmonized communications equipment operating in the 5 GHz frequency range intended for road safety and traffic management, and for non-safety related ITS applications; System Reference Document*. Technical Requirement. European Telecommunication Standard Institute, 2006 (cit. on p. 5).

[15] *5GAA - Coexistence of C-V2X and ITS-G5 at 5.9GHz*. White Paper. 5G Automotive Association, 2018 (cit. on p. 6).

[16] *IEEE 1609.0-2013 - IEEE Guide for Wireless Access in Vehicular Environments (WAVE) - Architecture*. Standard. Institute of Electrical and Electronics Engineers, 2014 (cit. on pp. 6–8).

[17] *IEEE 1609.2-2016 (Revision of IEEE Std 1609.2-2013) - IEEE Standard for Wireless Access in Vehicular Environments–Security Services for Applications and Management Messages*. Standard. Institute of Electrical and Electronics Engineers, 2016 (cit. on p. 6).

[18] *IEEE 1609.3-2016 (Revision of IEEE Std 1609.3-2010) - IEEE Standard for Wireless Access in Vehicular Environments (WAVE) – Networking Services*. Standard. Institute of Electrical and Electronics Engineers, 2016 (cit. on p. 6).

[19] *IEEE 1609.4-2016 (Revision of IEEE Std 1609.4-2010) - IEEE Standard for Wireless Access in Vehicular Environments (WAVE) – Multi-Channel Operation*. Standard. Institute of Electrical and Electronics Engineers, 2016 (cit. on pp. 6, 8).

[20] Syed Faraz Hasan. *Intelligent Transportation Systems*. eng. Springer International Publishing, 2018. ISBN: 3-319-64056-9 (cit. on p. 9).

[21] *IEEE 802.11-2016 - IEEE Standard for Information technology—Telecommunications and information exchange between systems Local and metropolitan area networks—Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*. Standard. Institute of Electrical and Electronics Engineers, 2016 (cit. on p. 9).

[22] *ETSI ES 202 663 V1.1.0 - Intelligent Transport Systems (ITS); European profile standard for the physical and medium access control layer of Intelligent Transport Systems operating in the 5 GHz frequency band*. Standard. European Telecommunication Standard Institute, 2010 (cit. on pp. 9, 10).

[23] *ETSI EN 303 613 V1.1.1 - Intelligent Transport Systems (ITS); LTE-V2X Access layer specification for Intelligent Transport Systems operating in the 5 GHz frequency band*. Standard. European Telecommunication Standard Institute, 2020 (cit. on pp. 9, 13).

[24] *ETSI EN 302 665 V1.1.1 - Intelligent Transport Systems (ITS); Communications Architecture*. Standard. European Telecommunication Standard Institute, 2010 (cit. on pp. 10–12).

[25] *ETSI TS 101 539-1 V1.1.1 - Intelligent Transport Systems (ITS); V2X Applications; Part 1: Road Hazard Signalling (RHS) application requirements specification*. Technical Specification. European Telecommunication Standard Institute, 2013 (cit. on pp. 10, 11).

[26] *ETSI TS 101 539-2 V1.1.1 - Intelligent Transport Systems (ITS); V2X Applications; Part 2: Intersection Collision Risk Warning (ICRW) application requirements specification*. Technical Specification. European Telecommunication Standard Institute, 2018 (cit. on pp. 10, 11).

[27] *ETSI TS 101 539-3 V1.1.1 - Intelligent Transport Systems (ITS); V2X Applications; Part 3: Longitudinal Collision Risk Warning (LCRW) application requirements specification*. Technical Specification. European Telecommunication Standard Institute, 2013 (cit. on p. 10).

[28] *ETSI TS 102 637-1 V1.1.1 - Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Part 1: Functional Requirements*. Technical Specification. European Telecommunication Standard Institute, 2010 (cit. on pp. 10, 11).

[29] *ETSI EN 302 637-2 V1.4.1 - Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Part 2: Specification of Cooperative Awareness Basic Service*. Standard. European Telecommunication Standard Institute, 2019 (cit. on pp. 10, 12).

[30] *ETSI EN 302 637-3 V1.3.1 - Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Part 3: Specifications of Decentralized Environmental Notification Basic Service*. Standard. European Telecommunication Standard Institute, 2019 (cit. on pp. 10, 12).

[31] *ETSI EN 302 636-1 V1.2.1 - Intelligent Transport Systems (ITS); Vehicular Communications; GeoNetworking; Part 1: Requirements*. Standard. European Telecommunication Standard Institute, 2014 (cit. on pp. 10, 20).

[32] *ETSI EN 302 636-2 V1.2.1 - Intelligent Transport Systems (ITS); Vehicular Communications; GeoNetworking; Part 2: Scenarios*. Standard. European Telecommunication Standard Institute, 2013 (cit. on pp. 10, 21).

[33] *ETSI EN 302 636-3 V1.1.2 - Intelligent Transport Systems (ITS); Vehicular Communications; GeoNetworking; Part 3: Network Architecture*. Standard. European Telecommunication Standard Institute, 2014 (cit. on pp. 10–12, 23).

[34] *ETSI EN 302 636-4-1 V1.4.1 - Intelligent Transport Systems (ITS); Vehicular Communications; GeoNetworking; Part 4: Geographical addressing and forwarding for point-to-point and point-to-multipoint communications; Sub-part 1: Media-Independent Functionality*. Standard. European Telecommunication Standard Institute, 2019 (cit. on pp. 10, 18, 26–29, 87–89).

[35] *ETSI EN 302 636-5-1 - Intelligent Transport Systems (ITS); Vehicular Communications; GeoNetworking; Part 5: Transport Protocols; Sub-part 1: Basic Transport Protocol*. Standard. European Telecommunication Standard Institute, 2017 (cit. on pp. 10, 16, 18, 26, 85, 86).

[36] *ETSI TS 102 687 V1.2.1 - Intelligent Transport Systems (ITS); Decentralized Congestion Control Mechanisms for Intelligent Transport Systems operating in the 5 GHz range; Access layer part.* Technical Specification. European Telecommunication Standard Institute, 2018 (cit. on pp. 10, 13).

[37] *Intelligent Transport Systems (ITS); Cross Layer DCC Management Entity for operation in the ITS G5A and ITS G5B medium.* Technical Specification. 2015 (cit. on pp. 10, 13).

[38] *3GPP TR 21.914 V14.0.0 - 3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; Release 14 Description; Summary of Rel-14 Work Items (Release 14).* Technical Requirement. 3rd Generation Partnership Project, 2018 (cit. on pp. 13, 14).

[39] *3GPP TR 21.915 V15.0.0 - 3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; Release 15 Description; Summary of Rel-15 Work Items (Release 15).* Technical Requirement. 3rd Generation Partnership Project, 2019 (cit. on p. 14).

[40] *Intelligent Transport Systems (ITS); GeoNetworking; Port Numbers for the Basic Transport Protocol (BTP).* Technical Specification. European Telecommunication Standard Institute, 2018 (cit. on p. 19).

[41] *Draft ETSI EN 302 636-6-1 V1.2.0 - Intelligent Transport Systems (ITS); Vehicular Communications; GeoNetworking; Part 6: Internet Integration; Subpart 1: Transmission of IPv6 Packets over GeoNetworking Protocols.* Standard. European Telecommunication Standard Institute, 2013 (cit. on p. 26).

[42] *ETSI TS 102 894-2 V1.2.1 - Intelligent Transport Systems (ITS); Users and applications requirements; Part 2: Applications and facilities layer common data dictionary.* Technical Specification. European Telecommunication Standard Institute, 2014 (cit. on pp. 26, 49).

[43] *ETSI EN 302 663 V1.2.0 - Intelligent Transport Systems (ITS); Access layer specification for Intelligent Transport Systems operating in the 5 GHz frequency band.* Standard. European Telecommunication Standard Institute, 2012 (cit. on pp. 26, 28).

[44] *ETSI TS 102 723-8 V1.1.1 - Intelligent Transport Systems (ITS); OSI cross-layer topics; Part 8: Interface between security entity and network and transport layer.* Technical Specification. European Telecommunication Standard Institute, 2016 (cit. on pp. 27, 61).

[45] *ETSI TS 102 636-4-2 V1.1.1 - Intelligent Transport Systems (ITS); Vehicular Communications; GeoNetworking; Part 4: Geographical addressing and forwarding for point-to-point and point-to-multipoint communications; Subpart 2: Media-dependent functionalities for ITS-G5.* Technical Specification. European Telecommunication Standard Institute, 2013 (cit. on p. 32).

[46] *Draft ETSI EN 302 931 V1.0.0 - Intelligent Transport Systems (ITS); Vehicular Communications; Geographical Area Definition*. Standard. European Telecommunication Standard Institute, 2010 (cit. on pp. 33, 43).

[47] M. Malinverno, F. Raviglione, C. Casetti, C. F. Chiasserini, J. Mangues-Bafalluy, and M. Requena-Esteso. «A Multi-Stack Simulation Framework for Vehicular Applications Testing». In: *Proceedings of the 10th ACM Symposium on Design and Analysis of Intelligent Vehicular Networks and Applications*. DIVANet '20. Alicante, Spain: Association for Computing Machinery, 2020, pp. 17–24. ISBN: 9781450381215. DOI: 10.1145/3416014.3424603. URL: https://doi.org/10.1145/3416014.3424603 (cit. on pp. 36–38, 44, 46, 49).

[48] N. Baldo, M. Miozzo, M. Requena-Esteso, and J. Nin-Guerrero. «An Open Source Product-oriented LTE Network Simulator Based on ns-3». In: *Proceedings of the 14th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*. 2011, pp. 293–298. URL: http://doi.acm.org/10.1145/2068897.2068948 (cit. on pp. 36, 45).

[49] F. Eckermann, M. Kahlert, and C. Wietfeld. «Performance Analysis of C-V2X Mode 4 Communication Introducing an Open-Source C-V2X Simulator». In: *2019 IEEE 90th Vehicular Technology Conference (VTC2019-Fall)*. 2019, pp. 1–5. DOI: 10.1109/VTCFall.2019.8891534 (cit. on pp. 37, 46).

[50] *About Eclipse SUMO*. URL: https://www.eclipse.org/sumo/about/ (cit. on p. 37).

[51] A. Wegener, M. Piorkowski, M. Raya, H. Hellbrück, S. Fischer, and J. Hubaux. «TraCI: An Interface for Coupling Road Traffic and Network Simulators». In: *Proceedings of the 11th Communications and Networking Simulation Symposium, CNS'08* (2008). DOI: 10.1145/1400713.1400740 (cit. on p. 37).